Towards Interactive Photorealistic Rendering

Dal Corso, Alessandro

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Towards Interactive Photorealistic Rendering

Alessandro Dal Corso

Kongens Lyngby 2018

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

七転び八起き.
Fall down seven times, stand up eight.
- Japanese proverb

Summary

Interactive rendering applications are becoming more and more prominent in everyday life. In many fields, including manufacturing, product design and entertainment, photorealistic rendering is useful in predicting the appearance of complex materials. However, due to production and time constraints, applications need to be interactive to provide immediate feedback to the user.

In this thesis, we address this challenge by proposing new photorealistic interactive rendering techniques, that leverage the parallel power of graphics processing units (GPUs) in order to effectively create renderings based on the laws of physics. These techniques propose effective caching and filtering schemes in order to efficiently reuse data, either across space or across time.

We provide insights into different areas of computer graphics, including scene reconstruction, material parameter estimation, efficient data structures and physically based rendering models. Our goal is to explore the different compromises and trade-offs that are necessary to achieve accurate photorealistic renderings. More specifically, we contribute with two techniques: the first relates to fast rendering of translucent materials, accounting for directional effects of subsurface scattering. The second technique contributes with a fast reprojection scheme to improve temporal stability in interactive ray tracing, that can be applied on top of existing rendering algorithms. On top of these, we propose an innovative validation pipeline to compare renderings with actual images, with the final purpose of validating existing rendering and reconstruction techniques against a picture of the real world.

With these contributions, we demonstrate how it is possible to use effective caching schemes to effectively improve existing techniques to handle more com-

plex optical effects, maintaining the time constraints of interactive rendering environments.

Summary (Danish)

Anvendelse af interaktive renderingsprogrammer bliver mere og mere fremtrædende i hverdagslivet. På mange områder, såsom produktfremstilling, produktdesign og underholdningsindustrien, bruges fotorealistisk rendering til forudsigelse af komplekse materialers udseende. Dog er der, pga. produktions- og tidsbegrænsninger, behov for interaktive programmer fordi brugere har brug for umiddelbar feedback.

I denne afhandling adresserer vi omtalte udfordring ved at foreslå nye interaktive fotorealistiske renderingsteknikker som bygger på grafikkortets (GPU'ens) egenskaber i forhold til parallelprocessering og derfor effektivt kan skabe renderinger der bygger på fysikkens love. Disse teknikker indeholder forslag mht. effektive caching- og filtreringsprocedurer til effektiv genbrug af data over rum eller over tid.

Vi giver indsigt i forskellige områder af computergrafik, herunder scene rekonstruktion, materiale parameterestimering, effektive datastrukturer og fysisk baserede renderingsmodeller. Vores mål er at udforske de forskellige kompromiser og afvejninger som er nødvendige for at opnå nøjagtige fotorealistiske renderinger. Mere specifikt bidrager vi med 2 teknikker: Den første er relateret til hurtig rendering af halvgennemsigtige materialer hvor der tages højde for retningsbestemte effekter i lysspredningen under overfladen. Den anden teknik er et bidrag i form af en hurtig genprojiceringsprocedure til forbedring af stabilitet over tid i interaktiv strålesporring (ray tracing). Dette kan anvendes som overbygning på eksisterende renderingsalgoritmer. Derudover bibringer vi også en innovativ valideringspipeline til sammenligning af renderede billeder med egentlige billeder. Hensigten er validering af eksisterende renderings- og rekonstruktionsteknikker

op imod et billede af den virkelige verden.

Med disse bidrag demonstrerer vi hvordan det er muligt at anvende effektive caching procedurer til effektivt at forbedre eksisterende teknikker til at kunne håndtere mere komplekse optiske effekter og stadig leve op til tidsbegrænsningerne i et interaktivt renderingsmiljø.

Preface

The work from this Ph.D. thesis was carried under the Section for Image Analysis and Computer Graphics, at the Department for Applied Mathematics and Computer Science (DTU Compute) at the Technical University of Denmark (DTU). This thesis was prepared in fulfillment of the requirements to obtain a doctor of philosophy (Ph.D.) in computer science. The work presented was funded on a DTU Compute internal scholarship.

This thesis deals with the topic of bringing techniques from photorealistic offline rendering into the interactive and real-time domain. To achieve this goal, a set of publications was produced over the course of the studies. The publications are summarized at page xi. Relevant publications for this thesis are available for the sake of the reader in Appendices I-IX. From these, a total of five peer reviewed publications have been included in this thesis (I-V), while two (X-XI) have been left out as not relevant to the overall thesis topic. Moreover, during the course of the Ph.D. some notes were produced. These notes are reported as non peer reviewed material and attached in Appendices VI-IX.

The work has been carried under the main supervision of Associate Professor Jeppe Eliot Revall Frisvad, with the co-supervision of Associate Professor Andreas Bærentzen. The research activities have mostly been conducted at DTU, with the exception of two external research stay periods. The first research stay was an internship at NVIDIA Corporation, under managers Aaron Lefohn and David Luebke, in the real-time rendering research team based in Redmond, Washington, USA. The second external research stay was under the supervision of Professor Toshiya Hachisuka, Department of Creative Informatics, The University of Tokyo.

Lyngby, 31-March-2018

Alessandro Dal Corso

Acknowledgements

First and foremost, I would like to thank my supervisors Jeppe Revall Frisvad and Andreas Bærentzen. In particular, I would like to thank Jeppe for the support during the Ph.D., the deep technical discussions and a push to strive for perfection. I would like to thank Andreas for the deep and interesting discussion on how to push my work towards more and more challenging directions. I would also like to thank my managers and mentors during my internship at the real-time rendering research group in NVIDIA: Craig Kolb, Marco Salvi, Aaron Lefohn and David Luebke, for sharing some of their deep knowledge about graphics. Finally, I would like to thank Toshiya Hachisuka and its group for hosting me at Tokyo University, for the their hospitality and the interesting discussions.

I would like to thank DTU Compute for financing my scholarship and giving me the opportunity to pursue a Ph.D. in the realm of computer graphics.

A giant thank you goes to my colleagues, collaborators, and office mates at the section for Image Analysis and Computer Graphics, and across all the groups I have visited over my Ph.D. Thank you for the support, the coffee calls, and the general silliness that makes everyday a fun day to come to the office. It has been a great pleasure to be part of this section, sharing thoughts and ideas, and having interesting discussion over coffee.

Last but not least, I would like to thank my loving family and friends. Without your continuous support and help, I would not have been able to complete this endeavor.

List of contributions

Peer rewiewed

- **Contribution I**: **Alessandro Dal Corso**, Jeppe Revall Frisvad, Thomas Kim Kjeldsen, and Jakob Andreas Bærentzen. Interactive Appearance Prediction for Cloudy Beverages. In Reinhard Klein and Holly Rushmeier, editors, *Workshop on Material Appearance Modeling*. The Eurographics Association, 2016. [Dal Corso et al., 2016]

- **Contribution II**: Jonathan Dyssel Stets, **Alessandro Dal Corso**, Jannik Boll Nielsen, Rasmus Ahrenkiel Lyngby, Sebastian Hoppe Nesgaard Jensen, Jakob Wilm, Mads Brix Doest, Carsten Gundlach, Eythor Runar Eiriksson, Knut Conradsen, Anders Bjorholm Dahl, Jakob Andreas Bærentzen, Jeppe Revall Frisvad, and Henrik Aanæs. Scene reassembly after multimodal digitization and pipeline evaluation using photorealistic rendering. *Applied Optics*, 56(27):7679–7690, Sep 2017. [Stets et al., 2017]
  Demonstration video: `http://www.imm.dtu.dk/~jerf/papers/videos/reassembly.mp4`

- **Contribution III**: **Alessandro Dal Corso**, Jeppe Revall Frisvad, Jesper Mosegaard, and J. Andreas Bærentzen. Interactive directional subsurface scattering and transport of emergent light. *The Visual Computer*, 33(3):371–383, Mar 2017. [Dal Corso et al., 2017a]
  Demonstration video: `http://people.compute.dtu.dk/alcor/videos/dirsssgpu.mov`

- **Contribution IV**: **Alessandro Dal Corso**, Marco Salvi, Craig Kolb, Jeppe Revall Frisvad, Aaron Lefohn, and David Luebke. Interactive stable ray tracing. In *Proceedings of High Performance Graphics*, HPG '17, pages

1:1–1:10, New York, NY, USA, 2017. ACM. [Dal Corso et al., 2017b]
Videos referenced in the paper: http://people.compute.dtu.dk/alcor/videos/srt_videos.zip

- **Contribution V**: **Alessandro Dal Corso**, Jonathan Dyssel Stets, Andrea Luongo, Jannik Boll Nielsen, Jeppe Revall Frisvad, and Henrik Aanæs. Virtual reality inspection and painting with measured BRDFs. In *SIGGRAPH Asia 2017 VR Showcase*, SA '17, pages 15:1–15:2, New York, NY, USA, 2017. ACM. [Dal Corso et al., 2017c]
Demonstration video: http://people.compute.dtu.dk/jerf/papers/videos/vrbrdf.mp4

Unpublished research notes

- **Contribution VI**: **Alessandro Dal Corso, Jeppe Revall Frisvad, Thomas Kim Kjeldsen**. *Derivation of standard and directional dipole quantities.* Unpublished note.
- **Contribution VII**: **Alessandro Dal Corso, Jeppe Revall Frisvad**. *Point cloud method for rendering BSSRDFs.* Unpublished note.
- **Contribution VIII**: **Alessandro Dal Corso, Jeppe Revall Frisvad**. *On BSSRDF estimation.* Unpublished note.
- **Contribution IX**: **Alessandro Dal Corso**. *Camera space BSSRDF importance sampling.* Unpublished note.

Not included peer reviewed contributions

- **Contribution X**: **Alessandro Dal Corso**, Mikkel Olsen, Kasper Hornbak Steenstrup, Jakob Wilm, Sebastian Jensen, Rasmus Reinhold Paulsen, Eythor Eiríksson, Jannik Boll Nielsen, Jeppe Revall Frisvad, Gudmundur Einarsson, and Hans Martin Kjer. Virtualtable: A projection augmented reality game. In *SIGGRAPH Asia 2015 Posters*, SA '15, pages 40:1–40:1, New York, NY, USA, 2015. ACM. [Dal Corso et al., 2015]
- **Contribution XI**: Henrik Aanæs, Knut Conradsen, **Alessandro Dal Corso**, Anders Bjorholm Dahl, Alessio Del Bue, Mads Emil Brix Doest, Jeppe Revall Frisvad, Sebastian Hoppe Nesgaard Jensen, Jannik Boll Nielsen, Jonathan Dyssel Stets, et al. Our 3D vision datasets in the making. In CVPR Workshop: The Future of Datasets in Vision 2015, 2015. [Aanæs et al., 2015]

List of symbols

Symbol	Unit	Meaning
$\mathbf{x}$	various	3D vector.
$\vec{\omega}$	various	Normalized 3D vector.
$\mathcal{X}$	various	Functional operator.
Φ	[W]	Radiant flux.
I	$[\mathrm{W} \cdot \mathrm{sr}^{-1}]$	Radiant intensity.
E	$[\mathrm{W} \cdot \mathrm{m}^{-2}]$	Irradiance.
B	$[\mathrm{W} \cdot \mathrm{m}^{-2}]$	Radiosity.
L	$[\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]$	Radiance.
σ_a	$[\mathrm{m}^{-1}]$	Absorption coefficient.
σ_s	$[\mathrm{m}^{-1}]$	Scattering coefficient.
$\sigma'_s = \sigma_s(1-g)$	$[\mathrm{m}^{-1}]$	Reduced scattering coefficient.
$\sigma_t = \sigma_s + \sigma_a$	$[\mathrm{m}^{-1}]$	Extinction coefficient.
$\sigma'_t = \sigma'_s + \sigma_a$	$[\mathrm{m}^{-1}]$	Reduced extinction coefficient.
$\alpha = \sigma_s/\sigma_t$	[–]	Single scattering albedo.
$\alpha' = \sigma'_s/\sigma'_t$	[–]	Reduced single scattering albedo.
$D = 1/(3\sigma'_t)$	[m]	Diffusion coefficient.
$\sigma_{\mathrm{tr}} = \sqrt{\sigma_a/D}$	$[\mathrm{m}^{-1}]$	Effective transport coefficient.
g	[–]	Scattering asymmetry parameter.
η	[–]	Relative index of refraction.
p	$[\mathrm{sr}^{-1}]$	Scattering phase function.
T_r	[–]	Transmittance calculated at distance r.
f_r	$[\mathrm{sr}^{-1}]$	Bidirectional reflectance distribution function (BRDF).
S	$[\mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]$	Bidirectional scattering-surface reflectance distribution function (BSSRDF).

Symbol	Unit	Meaning
$\hat{I}_N$	various	Monte Carlo estimator for integral I after N samples.
$\vec{n}$	[−]	Normal to the surface.
θ	[rad]	Polar or zenith angle.
ϕ	[rad]	Azimuthal angle.

Contents

Summary iii

Summary (Danish) v

Preface vii

Acknowledgements ix

List of contributions xi

List of symbols xiii

Contents xvi

1 Introduction 1
- 1.1 Scope . . . 1
- 1.2 Motivation . . . 2
  - 1.2.1 Digital prototyping and 3D printing . . . 4
  - 1.2.2 Architectural visualization . . . 6
  - 1.2.3 Computer games . . . 6
- 1.3 Outcome . . . 7
- 1.4 Outline . . . 8

2 Background 9
- 2.1 Radiometry . . . 9
- 2.2 Scattering Media . . . 12
  - 2.2.1 The radiative transfer equation . . . 12
  - 2.2.2 The BSSRDF . . . 14
  - 2.2.3 Connecting BSSRDFs and the radiative transfer equation 15
  - 2.2.4 Global solution to the BSSRDF . . . 20
  - 2.2.5 Rendering scattering media . . . 21
  - 2.2.6 Analytical BSSRDF Models . . . 24
- 2.3 Non-participating media . . . 28
  - 2.3.1 Transparent media . . . 28

2.3.2 Opaque media . . . 28
2.4 Rendering techniques . . . 31
2.4.1 Rasterization . . . 32
2.4.2 Ray tracing . . . 34

3 Related work 37
3.1 Typical approximations in physically based rendering . . . 38
3.2 Rendering techniques . . . 39
3.2.1 Caching . . . 40
3.2.2 Pre-computation . . . 42
3.2.3 Filtering . . . 43

4 Contributions 45
4.1 Photorealistic rendering for appearance prediction and parameter estimation . . . 46
4.2 Interactive rendering of scattering media . . . 50
4.3 Interactive stable ray tracing . . . 52
4.4 Applying interactive photorealistic techniques . . . 54
4.5 Discussion . . . 55

5 Conclusion 57

I Interactive Appearance Prediction for Cloudy Beverages 61

II Scene reassembly after multimodal digitization and pipeline evaluation using photorealistic rendering 67

III Interactive Directional Subsurface Scattering and Transport of Emergent Light 81

IV Interactive Stable Ray Tracing 95

V Virtual Reality inspection and painting with measured BRDFs 107

VI Derivation of standard and directional dipole quantities 111

VII Point cloud method for rendering BSSRDFs 127

VIII On BSSRDF estimation 131

IX Camera space BSSRDF importance sampling 137

Bibliography 141

Chapter 1

Introduction

1.1 Scope

The main goal of this thesis is to introduce new techniques to bring movie production quality rendering into interactive applications. The goal is to achieve photorealistic appearance as close as possible to the real world, while retaining the possibility for the user to interact with the simulation. Interactive realistic rendering applications are relevant in many fields, including

- 3D digital modeling and 3D printing.
- Product development and visualization.
- Digital prototyping.
- Computer games and animation.

In recent years, many advances have been made to achieve photorealistic quality in the interactive applications listed above, often achieving results that are both perceptually and remarkably close to slower photorealistic rendering techniques (see Figure 1.1). This thesis contributes with various techniques, exploring the area between interactive and real-time rendering paradigms (e.g. rasterization

Figure 1.1: Some examples of interactive and non-interactive photorealistic rendering. Left, rendering of a forest obtained in Maya (courtesy Greg Zdunek). Right, real-time rendering in *Far Cry 4* (courtesy Ubisoft).

and interactive ray tracing) and photorealistic rendering techniques (e.g. path tracing, hierarchical point cloud methods), with the goal of applying photorealistic appearance models. As most modern rendering techniques, we exploit the high throughput of graphics processing units (GPUs). Our contributions exploit the long standing GPU hardware rasterization pipeline, but also focus on applications of the increasingly expanding field of interactive ray tracing.

1.2 Motivation

From the beginning of computer graphics, researchers have been striving to synthesize more and more realistic images. Nowadays, for still images and carefully crafted scenes, it is almost impossible to distinguish a synthetic image from an actual photograph. To achieve these photorealistic images, a great amount of hours must be spent in either modeling or acquiring the 3D geometry, and either choosing or measuring the correct materials. Finally, some time must be spent rendering the image, that can take up to several hours. In the rendering part, the more advanced the physical model used, the higher the rendering time. This high rendering time can cause production bottlenecks, e.g. when some artistic tweaking is needed to meet a specific style or artistic constraint. Once the tweaks are performed, the whole rendering needs to be redone in order to see the

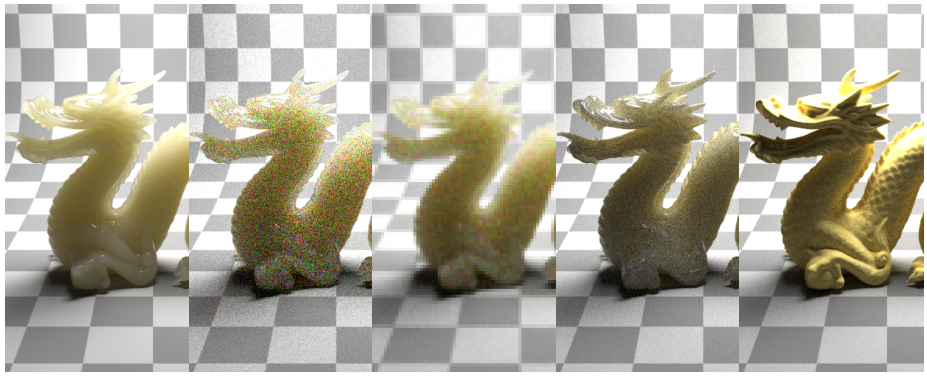


Figure 1.2: Comparing different preview techniques on a dragon rendered with white grapefuit juice material properties. From left to right: converged image, noisy unconverged image, downsampled image, different physical model, diffuse appearance. Preview images took 120 seconds to render on a GPU ray tracer. Converged result took 4 hours.

influence of the tweak on the final result, wasting precious production time.

To mitigate this problem, many applications display a preview of the final rendering result. Depending on the application the technique can be different (see Figure 1.2), like displaying a noisy result, a spatially downsampled one, or simply switching to a simpler rendering model. All these techniques come with different advantages and disadvantages. In general, these preview techniques trade physical accuracy in exchange of getting rid of noise.

Given these problems, when immediate feedback is important, there is a need in the industry for clever techniques that exploit the main two rendering constraints, namely time and accuracy. We classify and visualize various application areas categorized according to these constraints in Figure 1.3. The first constraint, allowed rendering time, ranges from a few milliseconds per frame in a real-time environment (e.g. games or virtual reality), to some fractions of a second (e.g. 3D printing preview) up to some minutes or hours (e.g. a frame in movie production environment). The second constraint, accuracy, describes the accuracy in modeling the physical process behind the rendering. In some applications, the required accuracy is limited. For example, in a video game, it is more important to obtain a fast believable result rather than a physically accurate one. In some other applications, the requirements are stricter: think architectural light simulation, where architects need to know precisely how the light reacts to different surfaces to see the overall illumination of the environ-

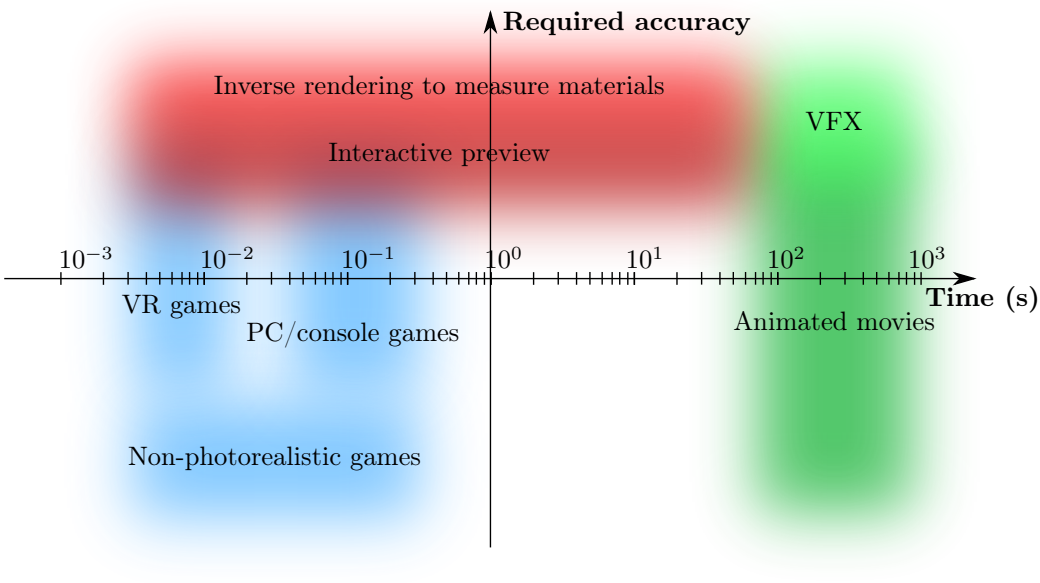


Figure 1.3: Visualizing the time-accuracy graph, showing the different application fields.

ment.

We now discuss three more detailed use cases, visualized in Figure 1.4, in which there is a demand for improvements in photorealistic rendering techniques. These cases are not meant to be comprehensive of all applications in which interactive photorealistic rendering techniques are needed. However, they provide insights on possible industrial applications of the various contributions within this thesis.

1.2.1 Digital prototyping and 3D printing

The first case we discuss is digital prototyping, in particular applied to 3D printing. In digital prototyping, we often want to preview the final appearance of a manufactured object. This is particularly hard in 3D printing, where the production process influences the shape and the appearance of the final object. Moreover, the materials used for 3D printing exhibit some particular radiometric properties, that makes them even more challenging to render accurately.

Figure 1.4: Applications of photorealistic interactive rendering. Left to right: preview in 3D printing application, architectural visualization (courtesy *Lumion*), virtual reality games (courtesy *The Climb* by *Crytek*).

In this application, the role of photorealistic rendering is dual. Apart from previewing the final result of the 3D printing process, we can actively use photorealistic rendering to measure optical properties of an object. In general, we start with limited or approximate knowledge of the optical properties of an object, that can be used to produce an initial rendering. We can then compare the rendering with a photograph of the object, in order to refine our estimate and get the actual optical properties. To enable this comparison, we require a specific setup and calibrated scene. We will explore this topic further in our Contributions I and II (see also Section 4.1). In both these applications, we require a high accuracy in our rendering (see Figure 1.3). In this case, a noisier or downsampled result can be acceptable, at the price of a certain inaccuracy in measuring the parameters.

All these innovations go towards a more strict integration of 3D printing processes and digital image synthesis as a whole (the digital-physical ecosystem). Photorealistic rendering helps reduce the number of iterations necessary to achieve a satisfying printed piece. This translates to reduced printing times for the same quality, and it avoids wasting potentially expensive printing material. Moreover, in a digital production environment, the validation through rendering enables us to perform quality assurance on the finished pieces, by comparing them with images of previously printed versions of the same piece.

1.2.2 Architectural visualization

Our second case is architectural visualization. In this particular field, architects use interactive visualization tools to show how light propagates through a building at different times of the day. This allows to create buildings that best use natural illumination, minimizing the overall electricity consumption. In this particular application, an important focus is placed on accurately represent materials. Materials used in construction like frosted glass, marble or cloth are radiometrically complex and hard to render fast and accurately. These materials can be either measured using some sort of apparatus, or matched by an artist so that they represent the appearance and light interaction properties of the original material. Given the complexity of the models and the generally needed artistic tweaking time, we find this field an interesting application field for interactive photorealistic techniques.

Recent efforts in this field have been trying to offer a more interactive experience, in order to show the finished product to project stakeholders. So, architectural visualization is moving towards interactive or real-time visualization, in particular in a virtual reality environment. Furniture giant IKEA is a particular example of this, where they use virtual and augmented reality to show how different pieces of furniture would fit in your own house. Virtual and augmented reality pipelines need to maintain the same standard of physically based accuracy as the original simulation, while fitting into extreme time constraints, a handful of milliseconds per frame. In this case, fast techniques that exploit current hardware to achieve photorealistic visualization are even more important.

1.2.3 Computer games

The final case we discuss is computer games. Computer games have the strictest time constraints of all the applications presented so far. For a standard frame in a game, the industry standard is around 16 milliseconds (60 frames per second). In recent years, virtual reality games are starting to become prominent, lowering the standard to 7-11 milliseconds total (90-120 frames per second). Given these constraints, most games use triangle based rasterization, so that they can leverage custom hardware on GPUs.

As games vary for gameplay, environment and style, photorealistic rendering can or cannot be employed in games. Nowadays, a good number of games strive to achieve a photorealistic or quasi-photorealistic look. In games, the focus is on performance and immersiveness, so the simulation does not need to be accurate from a physical standpoint. Sometimes, the rendering algorithms are made non

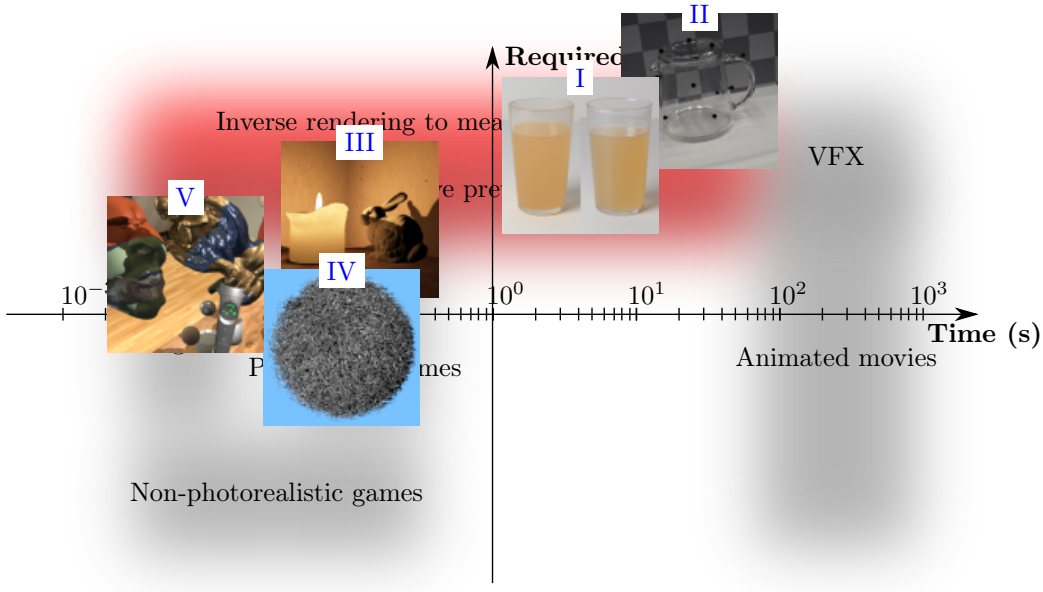


Figure 1.5: The main contributions from this thesis on the time-accuracy graph of Figure 1.3. The roman number on each contribution links to the original published text. A discussion on each individual contribution is available in Chapter 4.

realistic on purpose to achieve a specific artistic look. Given the hard time constraints, specific techniques need to be developed to achieve a photorealistic look. These techniques are often tailored to the characteristics of a specific game, and no overall solution for all games exists. This makes it an interesting field to contribute with physically based techniques, with the ultimate goal to reduce production costs.

1.3 Outcome

During the three years period, we achieved a number of academic publications, reported at page xi (some publications come with attached videos, linked in the same page). In these publications, we managed to achieve a number of insights in the realm of computer graphics and material appearance. In particular, we created a range of techniques that can be readily applied to existing interactive applications. We believe we pushed the boundary on the degree of realism that

can be achieved by modern GPUs on the same strict time constraints.

In particular, papers III and IV achieve this by providing widely applicable caching in the realm of interactive subsurface scattering and global illumination. Figure 1.5 shows how our contributions fit within the time-accuracy graph of Figure 1.3.

1.4 Outline

A list of all the contributions created during the course of the Ph.D. studies at page xi, including some unpublished notes, that we will refer mostly from Chapter 2 to provide additional details. All the various papers are available as appendices.

The thesis is divided into four main chapters. In Chapter 2, we first cover some background on some of the necessary foundations in radiometry, photorealistic appearance models and interactive rendering techniques. We will then cover in Chapter 3 an overarching literature review on the current efforts on photorealistic interactive rendering, referring to the individual contributions for more detailed related work. In Chapter 4, most importantly, we will go into more detail about the individual contributions and how each one of them contributes to reaching the outcome of the thesis. We will finally summarize our conclusions in Chapter 5.

Chapter 2

Background

In this section, we will introduce the theoretical elements necessary to complete the related work and the contributions of Chapters 3 and 4. Our contributions range a wide span of physically based materials, including both scattering and non scattering media. We will describe both in this chapter (Sections 2.2 and 2.3 respectively), with a brief introduction to basic radiometric concept (Section 2.1). We will finally conclude (Section 2.4) with a more detailed description of the two main rendering paradigms we used to achieve our results, rasterization and ray tracing. A table with relevant notation used in this chapter is available at page xiii.

2.1 Radiometry

Radiometry is a branch of physics that describes and measures electromagnetic radiation. In this section, we will define some basic radiometric quantities. In particular, we want to give a definition of radiance, the most useful quantity in describing light transport along rays.

First, let us consider an ideal point source of photons in space. This source emits a certain amount of energy U, measured in Joules $[J]$. The first quantity

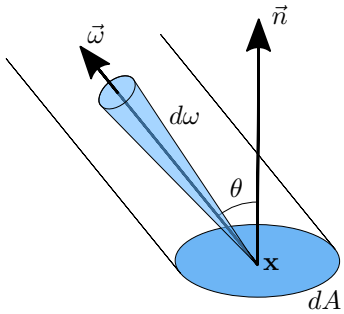


Figure 2.1: Configuration to define radiance.

we derive is radiant flux or radiant power Φ, defined as the amount of energy per second emitted by the light:

$$\Phi = \frac{dU}{dt} \quad [\mathrm{W}].$$

The total flux represents the power the light emits in all directions. We usually want to be more descriptive on how a light or a surface is emitting light, since not all the sources we consider are ideal. First, we are interested on how light emission changes across directions. We define as *radiant intensity* I the amount of flux the light emits towards a specific direction $\vec{\omega}$:

$$I(\vec{\omega}) = \frac{d\Phi}{d\omega} \quad [\mathrm{W} \cdot \mathrm{sr}^{-1}],$$

where $d\omega$ is an infinitesimal solid angle centered on direction $\vec{\omega}$. An isotropic point light, by definition, has constant intensity across all directions.

We now consider an infinitesimal surface area element dA that receives light, with center point $\mathbf{x}$. We now define *irradiance* as the amount of incoming flux received per unit area:

$$E(\mathbf{x}) = \frac{d\Phi}{dA} \quad [\mathrm{W} \cdot \mathrm{m}^{-2}].$$

If the flux is emitted by the surface, we use the name *radiant exitance* and define it with the symbol M. The most simple irradiance is the one emitted by an isotropic point light. In this case, the irradiance at a distance R from the point light is

$$E = \frac{\Phi}{4\pi R^2} \quad [\mathrm{W} \cdot \mathrm{m}^{-2}].$$

So, the farther we go from a point light, the less irradiance we receive, because the power Φ is spread across a larger area.

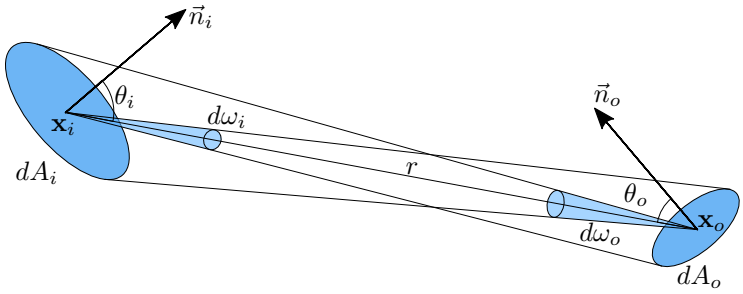


Figure 2.2: Configuration to prove the equality of radiance across a ray.

Finally, we combine the definitions of intensity and irradiance above into one, to define our last important basic radiometric quantity, *radiance*:

$$L(\mathbf{x}, \vec{\omega}) = \frac{d^2\Phi}{dA \cos\theta d\omega} \quad [\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}],$$

where $\cos\theta = \vec{n} \cdot \vec{\omega}$ is the cosine of the angle between the normal at $\mathbf{x}$ and the direction of evaluation. $dA\cos\theta$ is called the *projected area element*. As irradiance, radiance can be incoming or outgoing from a specific point. We indicate these quantities with L_i and L_o, respectively. In graphics, radiance is usually the most useful quantity for two main reasons. First, the other quantities can be easily computed from radiance through radiometric integrals:

$$\begin{aligned}
\Phi &= \int_{\Omega^+} \int_A L(\mathbf{x}, \vec{\omega})(\vec{n} \cdot \vec{\omega})\ dA\ d\omega, \\
I(\vec{\omega}) &= \int_A L(\mathbf{x}, \vec{\omega})(\vec{n} \cdot \vec{\omega})\ dA, \\
E(\mathbf{x}) &= \int_{\Omega^+} L(\mathbf{x}, \vec{\omega})(\vec{n} \cdot \vec{\omega})\ d\omega,
\end{aligned}$$

where Ω^+ and A are the hemisphere around $\vec{n}$ and the total surface, respectively. Second, radiance carried by a ray *in vacuo* is constant. We can prove this quite easily, with the aid of Figure 2.2. Note that for point $\mathbf{x}_o$, we have by definition $d\omega_o = \frac{dA_i \cos\theta_i}{r^2}$, and similarly $d\omega_i = \frac{dA_o \cos\theta_o}{r^2}$. Then:

$$\begin{aligned}
L_i(\mathbf{x}_i, \vec{\omega}_i) = \frac{d^2\Phi}{dA_i \cos\theta_i d\omega_i} &= \frac{d^2\Phi}{dA_i \cos\theta_i \frac{dA_o \cos\theta_o}{r^2}} \\
&= \frac{d^2\Phi}{\frac{dA_i \cos\theta_i}{r^2} dA_o \cos\theta_o} = \frac{d^2\Phi}{dA_o \cos\theta_o d\omega_o} = L_o(\mathbf{x}_o, \vec{\omega}_o).
\end{aligned}$$

Note that we assume that the flux does not vary across the path, which is generally true if no objects are in the way and there is no medium in between the two points causing scattering events.

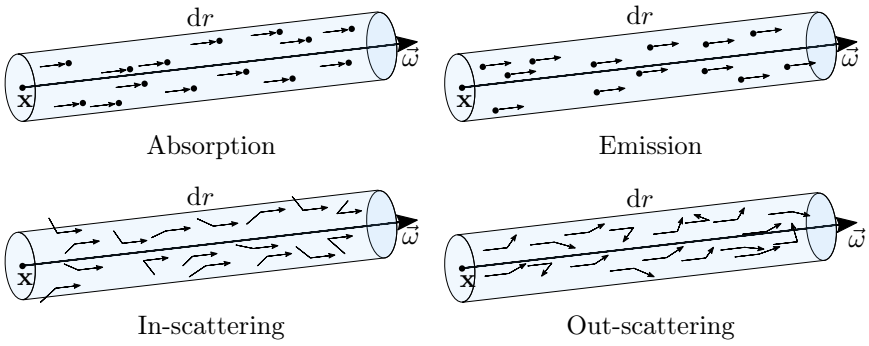


Figure 2.3: Individual processes in the local form of the radiative transfer equation: absorption, emission, in-scattering and out-scattering. The directional derivative $\vec{\omega}\cdot\nabla L(\mathbf{x},\vec{\omega})$ corresponds on the variation over an infinitesimal length element $\mathrm{d}r$ in direction $\vec{\omega}$.

2.2 Scattering Media

The first group of physically based materials we tackle is scattering media. We deal with efficient rendering of scattering media in Contributions I and III, so we introduce relevant theory to better contextualize those contributions.

Scattering is the process photons go through once they hit an optically dense medium. Photons are deflected from their current path by electromagnetic forces, due to atom nuclei and atomic bonds. It is possible to model this process mathematically. We start by discussing one of such models, the radiative transfer equation, which is a local solution of the scattering process for a point in the medium. Then, we will provide a more global formulation of the scattering process as a function of light propagating between two surface points, the BSSRDF. We will then proceed to prove that the BSSRDF indeed respects the radiative transfer equation, and then describe two ways to render materials: one using the radiative transfer equation, one using BSSRDF. Finally, we will introduce three BSSRDF analytical models that can be used in rendering.

2.2.1 The radiative transfer equation

The radiative transfer equation [Chandrasekhar, 1950] describes light propagation in scattering media. It is a particular form of the Boltzmann transport equation for a thermodynamic system. In this section, we give its integro-differential form, that describes how radiance $L(\mathbf{x},\vec{\omega})$ varies at a point $\mathbf{x}$ in a

scattering medium towards direction $\vec{\omega}$. For the sake of simplicity, from now on we assume a scattering media that respects linear optics (excluding i.e. fluorescent materials), and in the stationary state (i.e. the radiance L is not changing in time).

The radiative transfer equation describes light traveling a medium as subject to four processes: emission, absorption, out-scattering and in-scattering. Each of these processes can be expressed in terms the directional derivative $\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega})$ $[\mathrm{W} \cdot \mathrm{m}^{-3} \cdot \mathrm{sr}^{-1}]$. All effects are illustrated in Figure 2.3.

Emission increases the overall radiance of the ray by a term $q(\mathbf{x}, \vec{\omega})$:

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = q(\mathbf{x}, \vec{\omega}) \quad [\mathrm{W} \cdot \mathrm{m}^{-3} \cdot \mathrm{sr}^{-1}].$$

Emission describes how some form of energy (like heat) is becoming radiant energy.

Absorption is the dual effect of emission, where photons are absorbed by the material, and generally being transformed from radiant energy into heat energy. On the opposite hand of emission, the loss due to absorption is proportional to the radiance at the point L. The *absorption coefficient* $\sigma_a(\mathbf{x}, \vec{\omega})[\mathrm{m}^{-1}]$ describes the amount of absorption per unit traveled within the medium. In formulas,

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = -\sigma_a(\mathbf{x}, \vec{\omega}) L(\mathbf{x}, \vec{\omega}) \quad [\mathrm{W} \cdot \mathrm{m}^{-3} \cdot \mathrm{sr}^{-1}].$$

Out-scattering describes photons that are deflected from their original path $\vec{\omega}$ due to interaction with the atoms of the material. The effect is similar to absorption, with a different coefficient named the *scattering coefficient* $\sigma_s(\mathbf{x}, \vec{\omega})[\mathrm{m}^{-1}]$. This gives a loss of radiance that is similar to absorption:

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = -\sigma_s(\mathbf{x}, \vec{\omega}) L(\mathbf{x}, \vec{\omega}) \quad [\mathrm{W} \cdot \mathrm{m}^{-3} \cdot \mathrm{sr}^{-1}].$$

Absorption and out-scattering, given their similarities, are often combined into one effect, *attenuation*, described by a value called the extinction coefficient $\sigma_t(\mathbf{x}, \vec{\omega}) = \sigma_s(\mathbf{x}, \vec{\omega}) + \sigma_a(\mathbf{x}, \vec{\omega})[\mathrm{m}^{-1}]$.

In-scattering, finally, describes the radiance aligning towards $\vec{\omega}$ from other scattering events. Let us consider another direction $\vec{\omega}'$ from $\mathbf{x}$. We define a probability distribution for a photon to scatter from $\vec{\omega}'$ towards an infinitesimal solid angle $d\omega$ around $\vec{\omega}$. This probability distribution is called the *phase function* $p(\mathbf{x}, \vec{\omega}', \vec{\omega})[\mathrm{sr}^{-1}]$. With the phase function, and integrating over all directions, we get the effect of in-scattering.

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = \sigma_s(\mathbf{x}, \vec{\omega}) \int_{4\pi} L(\mathbf{x}, \vec{\omega}') p(\mathbf{x}, \vec{\omega}', \vec{\omega}) d\omega' \quad [\mathrm{W} \cdot \mathrm{m}^{-3} \cdot \mathrm{sr}^{-1}].$$

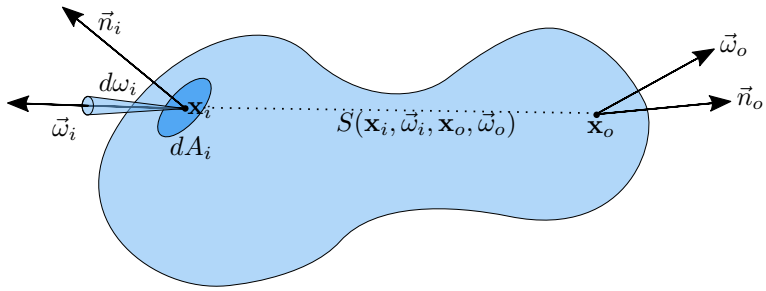


Figure 2.4: Configuration in which we define the BSSRDF for a generic surface.

Note the multiplication by σ_s to account for the right amount of scattering as we move along the ray. From the phase function, we can calculate the dimensionless radiometric property $g(\mathbf{x}, \vec{\omega})$, describing the mean cosine of the scattering angle between $\vec{\omega}$ and $\vec{\omega}'$ as

$$g(\mathbf{x}, \vec{\omega}) = \int_{4\pi} (\vec{\omega}' \cdot \vec{\omega}) p(\mathbf{x}, \vec{\omega}', \omega) d\omega' \quad [-].$$

As we will see, most analytical models describe a scattering medium in terms of the three coefficients σ_s, σ_a and g, plus the ratio of the index of refraction inside the medium and the one outside the medium η.

We can now combine all terms to obtain the complete integro-differential form of the radiative transfer equation:

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = q(\mathbf{x}, \vec{\omega}) - \sigma_t(\mathbf{x}, \vec{\omega}) L(\mathbf{x}, \vec{\omega}) + \sigma_s(\mathbf{x}, \vec{\omega}) \int_{4\pi} L(\mathbf{x}, \vec{\omega}') p(\mathbf{x}, \vec{\omega}', \vec{\omega}) d\omega'. \tag{2.1}$$

2.2.2 The BSSRDF

In Section 2.1, we have defined radiometric quantities as either incoming or outgoing. We will now describe how radiometric quantities change between incoming and outgoing. This gives a way to describe light interaction due to a surface.

Let us first consider a surface A illuminated by a light, as in Figure 2.9. Let us consider the element of flux $d\Phi_i$ arriving from direction $\vec{\omega}_i$ on a surface element dA_i centered at a point $\mathbf{x}_i$. Due to surface interaction, part of the incoming

light will emerge at a point $\mathbf{x}_o$, in direction $\vec{\omega}_o$. We consider the proportionality factor between the emitted radiance and the incoming flux:

$$dL(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) d\Phi_i(\mathbf{x}_i, \vec{\omega}_i) \quad [\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]. \tag{2.2}$$

The factor S, dependent on the surface, is called *bidirectional scattering-surface reflectance distribution function*, or BSSRDF for short [Nicodemus et al., 1977]. The units for the BSSRDF are $[\mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]$. From the definition of BSSRDF and flux, we obtain right away the reflected radiance equation:

$$L(\mathbf{x}_o, \vec{\omega}_o) = \int_A \int_{\Omega^+} S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_i, \vec{\omega}_i) (\vec{n}_i \cdot \vec{\omega}_i) d\omega_i dA_i \quad [\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]. \tag{2.3}$$

Note that we did not assume anything about the material, deriving the BSSRDF from purely radiometric quantities. The BSSRDF can be seen as a global formulation of scattering processes within the material.

Note that Equation 2.3 does not give the full outgoing radiance distribution L_o, since we need to add the radiance emitted by the body:

$$L_o(\mathbf{x}, \vec{\omega}) = L_e(\mathbf{x}, \vec{\omega}) + L(\mathbf{x}, \vec{\omega}).$$

This equation gives the so called extended form of the rendering equation [Jensen et al., 2001]. Given the physical nature of the BSSRDF, we can impose conservation of energy, so that the reflected light never increases:

$$0 \leq \int_A \int_{\Omega} S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) (\vec{n}_i \cdot \vec{\omega}_i) d\omega_i dA_i \leq 1 \quad [-]. \tag{2.4}$$

Moreover, the BSSRDF respects the Stokes-Helmholtz reciprocity principle [Stokes, 2009; Chandrasekhar, 1958]:

$$S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = S(\mathbf{x}_o, \vec{\omega}_o, \mathbf{x}_i, \vec{\omega}_i) \quad [\mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]. \tag{2.5}$$

2.2.3 Connecting BSSRDFs and the radiative transfer equation

Given the above definition of BSSRDF (Equation 2.2), one may wonder how the BSSRDF is related to any scattering process at all, since it simply describes a relationship between an incoming and an outgoing radiometric quantity. In this section, starting from the definition of the BSSRDF, we will derive the local integro-differential form of the radiative transfer Equation 2.1, to show how the BSSRDF fits nicely as a mathematical description of an underlying scattering

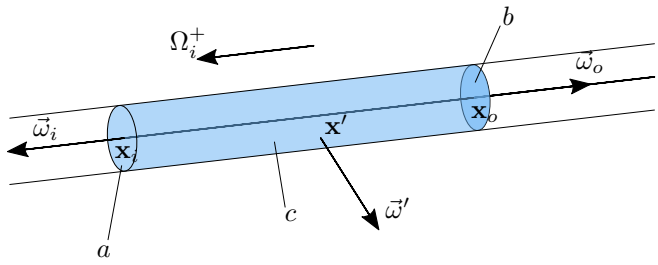


Figure 2.5: Cylinder configuration to prove the radiance solution.

process. Most of this derivation comes from Preisendorfer [1965, 1976], where the BSSRDF is called *scattering function*.

To show this relationship, we define some mathematical operators. An operator in our context maps a function into another function, as does this $\mathcal{Q}$ to integrate over an hemisphere:

$$\mathcal{Q} = \int_{\Omega} [\ ] d\omega.$$

So, using the operator left associativity, $E = L\mathcal{Q}$ denotes the equation

$$E(\mathbf{x}) = \int_{\Omega} L(\mathbf{x}, \vec{\omega}) d\omega.$$

Note that in this case operator $\mathcal{Q}$ maps the functional L into functional E. Where obvious, we drop the dependencies on $\mathbf{x}$ and ω from the operator for the sake of clarity. We can now define a new functional operator $\mathcal{Z}$, also called the *standard operator*:

$$\mathcal{Z}(a,b) = \int_a \int_{\Omega_i^+} [\ ] S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o)(\vec{n}_i \cdot \vec{\omega_i}) d\omega_i dA_i.$$

Where S is the BSSRDF, a and b are parts of the surface of the medium containing $\mathbf{x}_i$ and $\mathbf{x}_o$, respectively, and Ω_i^+ is the hemisphere oriented towards $\vec{n}_i$. This allows us to write $L = L_i \mathcal{Z}(a,b)$ for equation 2.3, greatly simplifying notation. We want now to describe how radiance varies across a path. Let us consider the configuration of Figure 2.5. In this figure, we have a path $P_r(\mathbf{x}_i, \vec{\omega})$ starting at $\mathbf{x}_i$, with direction $\vec{\omega}_o$ for r units, terminating in $\mathbf{x}_o$ (so that $\mathbf{x}_o = \mathbf{x}_i + r\vec{\omega}_o$). Let us now consider a cylindrical scattering medium C with the same index of refraction as the surroundings, composed of three parts: a top circle a on $\mathbf{x}_i$, a bottom circle b on $\mathbf{x}_o$ and a flank surface c. We orient the cylinder so that the top a points towards $-\vec{\omega}_o$, and the center of the cylinder is aligned with $P_r(\mathbf{x}_i, \vec{\omega}_o)$. This defines a direction for the hemispheres, e.g. Ω_i^+ and Ω_i^- are the hemispheres centered in $\mathbf{x}_i$ and oriented towards or against $\vec{n}_i$, respectively.

As a final bit of notation, we indicate with $L^+(a)$ some radiance $L(\mathbf{x}, \vec{\omega})$ where $\mathbf{x} \in a$ and $\vec{\omega} \in \Omega^+$.

At the steady state and by conservation of energy, the total radiance going out of the cylinder through the surface b is:

$$L^-(b) = L^+(b)\mathcal{Z}(b,b) + L^-(a)\mathcal{Z}(a,b) + L^-(c)\mathcal{Z}(c,b). \tag{2.6}$$

We want now to find the behavior of the system at equilibrium, when the cylinder becomes thinner and thinner ($C \to P_r(\mathbf{x}_i, \vec{\omega}_o)$). We will analyze each one of the terms in Equation 2.6 independently.

The first term on the right hand side of Equation 2.6 describes the part of $L^-(b)$ that stays inside due to reflection on b. This can be shown to tend to zero as the cylinder shrinks. This comes from the fact that for a transparent plane, all the radiance contribution passes through so none is reflected back. In formulas,

$$\lim_{C \to P_r(\mathbf{x}_i, \vec{\omega}_o)} [L^+(b)\mathcal{Z}(b,b)](\mathbf{x}_o, \vec{\omega}_o) = 0. \tag{2.7}$$

The second term defines the radiance transmitted between a and b. As the cylinder shrink, the photons have less and less room to scatter within the cylinder, so only the photons staying directly on $\vec{\omega}$ (even if they scatter back and forth) will be considered at the end. Let us consider the limit:

$$L_r^0(\mathbf{x}_o, \vec{\omega}_o) = \lim_{C \to P_r(\mathbf{x}_i, \vec{\omega}_o)} [L^-(a)\mathcal{Z}(a,b)](\mathbf{x}_o, \vec{\omega}_o).$$

At the limit $a \to \mathbf{x}_i$, we have $L^-(a) = L^0(\mathbf{x}_i, \vec{\omega}_o)$, that can be brought out:

$$L_r^0(\mathbf{x}_o, \vec{\omega}_o) = L^0(\mathbf{x}_i, \vec{\omega}_o) \lim_{C \to P_r(\mathbf{x}_i, \vec{\omega}_o)} [\mathcal{Z}(a,b)](\mathbf{x}_o, \vec{\omega}_o) = L^0(\mathbf{x}_i, \vec{\omega}_o) T_r(\mathbf{x}_i, \vec{\omega}_o), \tag{2.8}$$

where the last term is called *beam transmittance*, i.e. the amount of light blocked on the direct path. We define a dual term $1 - T_r(\mathbf{x}_i, \vec{\omega})$, called *beam attenuation*. The rate of change of beam attenuation per unit length at $\mathbf{x}_i$ on $\vec{\omega}$ is the extinction coefficient defined in Section 2.2.1:

$$\sigma_t(\mathbf{x}_i, \vec{\omega}) = \lim_{r \to 0} \frac{1 - T_r(\mathbf{x}_i, \vec{\omega})}{r},$$

which leads to a natural definition for the beam transmittance

$$T_r(\mathbf{x}_i, \vec{\omega}) = \exp\left(- \int_0^r \sigma_t(\mathbf{x}_i + r'\vec{\omega}, \vec{\omega}) dr' \right).$$

We now need to calculate the last term in the summation, which we call L_r^*:

$$L_r^*(\mathbf{x}_o, \vec{\omega}_o) = \lim_{C \to P_r(\mathbf{x}_i, \vec{\omega}_o)} [L^-(c)\mathcal{Z}(c,b)](\mathbf{x}_o, \vec{\omega}).$$

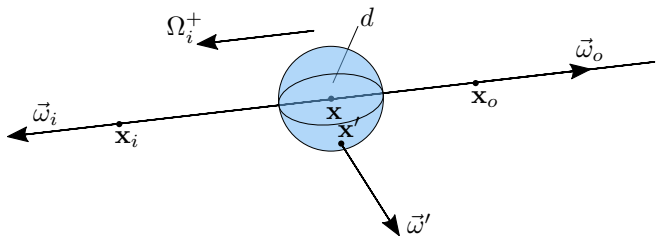


Figure 2.6: Sphere configuration to prove the radiance solution.

We will not include the full derivation, but by subdividing the cylinder into infinitesimally thin slices, each with its own transmittance, and then taking the limit, to obtain a integral form for L_r^*:

$$L_r^*(\mathbf{x}_o, \vec{\omega}_o) = \int_0^r L_*(\mathbf{x}', \vec{\omega}_o) T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr', \tag{2.9}$$

where $\mathbf{x}' = \mathbf{x}_i + r'\vec{\omega}_o$ and where L_* is the radiance per unit length

$$L_*(\mathbf{x}', \vec{\omega}_o) = \lim_{r \to 0} \frac{L_r^*(\mathbf{x}_o, \vec{\omega}_o)}{r}.$$

By putting together 2.7, 2.8 and 2.9 into 2.6, we obtain the following form:

$$L_r(\mathbf{x}_o, \vec{\omega}_o) = L_r^0(\mathbf{x}_o, \vec{\omega}_o) + L_r^*(\mathbf{x}_o, \vec{\omega}_o), \tag{2.10}$$

$$L_r(\mathbf{x}, \vec{\omega}_o) = L^0(\mathbf{x}_i, \vec{\omega}_o) T_r(\mathbf{x}_i, \vec{\omega}_o) + \int_0^r L_*(\mathbf{x}', \vec{\omega}_o) T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr'.$$

We only need now to define L_* in terms of L. For this derivation, let us still consider the configuration of Figure 2.6, where we have a point $\mathbf{x}'$ and vector $\vec{\omega}'$ on the surface of a sphere d surrounding a point $\mathbf{x}$ on path $P_r(\mathbf{x}_i, r)$ with direction $\vec{\omega} = \vec{\omega}_s$. It is possible to derive an approximation of S for small s as

$$S(\mathbf{x}', \vec{\omega}', \mathbf{x}, \vec{\omega}) = \sigma_s(\mathbf{x}', \vec{\omega}', \vec{\omega}) \frac{s}{A_i'} + o(s).$$

$\mathbf{x}', \mathbf{x}$ are two points on the sphere, s is the sphere radius, and A_i' is the area of the sphere that would be lit by a light shined from direction $-\vec{\omega}'$. The term $\sigma_s(\mathbf{x}', \vec{\omega}', \vec{\omega}) = \sigma_s(\mathbf{x}', \vec{\omega}') p(\mathbf{x}', \vec{\omega}', \vec{\omega})$ is the non-normalized phase function. $o(s)$ is an error such as $\lim_{s \to 0} \frac{o(s)}{s} = 0$. We can now recalculate L_s^* for the sphere configuration using the approximation:

$$\begin{aligned} L_s^*(\mathbf{x}, \vec{\omega}) &= \int_{A_i'} \int_{\Omega_i^-} L(\mathbf{x}', \vec{\omega}') [\sigma_s(\mathbf{x}', \vec{\omega}', \vec{\omega}) \frac{s}{A_i'} + o(s)] (\vec{n}' \cdot \vec{\omega}') d\omega' dA_i' \\ &= \int_{4\pi} L(\mathbf{x}', \vec{\omega}') [\sigma_s(\mathbf{x}', \vec{\omega}', \vec{\omega}) s + A_i' o(s)] (\vec{n}' \cdot \vec{\omega}') d\omega' \\ &= s \int_{4\pi} L(\mathbf{x}', \vec{\omega}') \sigma_s(\mathbf{x}', \vec{\omega}', \vec{\omega}) d\omega' + o(s) \int_{4\pi} L(\mathbf{x}', \vec{\omega}') A_i' d\omega'. \end{aligned} \tag{2.11}$$

As the sphere gets smaller $s \to 0$, the second term disappears with the $o(s)$, so that we finally obtain L_* as

$$L_*(\mathbf{x}, \vec{\omega}) = \lim_{s \to 0} \frac{L_s^*(\mathbf{x}, \vec{\omega})}{s} = \sigma_s(\mathbf{x}, \vec{\omega}) \int_{4\pi} L(\mathbf{x}, \vec{\omega}') p(\mathbf{x}, \vec{\omega}', \vec{\omega}) d\omega'. \tag{2.12}$$

Putting it all together, we obtain the so called *integral form* of the radiative transfer equation:

$$\begin{aligned} L_r(\mathbf{x}_o, \vec{\omega}_o) &= L^0(\mathbf{x}_i, \vec{\omega}_o) T_r(\mathbf{x}_i, \vec{\omega}_o) + \\ &+ \int_0^r \sigma_s(\mathbf{x}', \vec{\omega}_o) \int_{4\pi} L(\mathbf{x}', \vec{\omega}') p(\mathbf{x}', \vec{\omega}', \vec{\omega}_o) d\omega' T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr'. \end{aligned} \tag{2.13}$$

This form is described along a the extent of a path rather than a single point as in the integro-differential form in Equation 2.1. We now derive back that form by deriving the above across r, which is the same as a directional derivative $\vec{\omega} \cdot \nabla$. Of the two terms on the right hand side of the equation above, the first becomes:

$$\begin{aligned} \frac{d}{dr} L^0(\mathbf{x}_i, \vec{\omega}_o) T_r(\mathbf{x}_i, \vec{\omega}_o) &= L^0(\mathbf{x}_i, \vec{\omega}_o) \frac{d}{dr} T_r(\mathbf{x}_i, \vec{\omega}_o) \\ &= L^0(\mathbf{x}_i, \vec{\omega}_o)(-\sigma_t(\mathbf{x}_o, \vec{\omega}_o) T_r(\mathbf{x}_i, \vec{\omega}_o)) = -\sigma_t(\mathbf{x}_o, \vec{\omega}_o) L_r^0(\mathbf{x}_o, \vec{\omega}_o), \end{aligned}$$

where the last step comes from Equation 2.8. As for the second term, we use the form from Equation 2.12:

$$\begin{aligned} \frac{d}{dr} L_r^*(\mathbf{x}_o, \vec{\omega}_o) &= \frac{d}{dr} \int_0^r L_*(\mathbf{x}', \vec{\omega}_o) T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr' \\ &= \int_0^r L_*(\mathbf{x}', \vec{\omega}_o) \frac{d}{dr} T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr' + L_*(\mathbf{x}_o, \vec{\omega}_o) \\ &= -\sigma_t(\mathbf{x}, \vec{\omega}_o) \int_0^r L_*(\mathbf{x}', \vec{\omega}_o) T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr' + L_*(\mathbf{x}_o, \vec{\omega}_o) \\ &= -\sigma_t(\mathbf{x}_o, \vec{\omega}_o) L_r^*(\mathbf{x}_o, \vec{\omega}_o) + L_*(\mathbf{x}_o, \vec{\omega}_o). \end{aligned}$$

By putting it all together:

$$\begin{aligned} \frac{d}{dr} L_r(\mathbf{x}_o, \vec{\omega}_o) &= -\sigma_t(\mathbf{x}_o, \vec{\omega}_o)[L_r^0(\mathbf{x}, \vec{\omega}_o) + L_r^*(\mathbf{x}_o, \vec{\omega}_o)] + L_*(\mathbf{x}_o, \vec{\omega}_o) \\ &= -\sigma_t(\mathbf{x}_o, \vec{\omega_o}) L_r(\mathbf{x}, \vec{\omega}_o) + L_*(\mathbf{x}_o, \vec{\omega}_o). \end{aligned}$$

By dropping the dependency on r and the o subscript, and introducing the definition for L_* (Equation 2.12) and the directional derivative symbol, we obtain the integro-differential form of the radiative transfer equation

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = -\sigma_t(\mathbf{x}, \vec{\omega}) L(\mathbf{x}, \vec{\omega}) + \sigma_s(\mathbf{x}, \vec{\omega}) \int_{4\pi} L(\mathbf{x}, \vec{\omega}') p(\mathbf{x}, \vec{\omega}', \vec{\omega}) d\vec{\omega}'.$$

We can then add an additional term to account for emission, obtaining the form in Equation 2.1.

2.2.4 Global solution to the BSSRDF

Now, we proved that the BSSRDF and the RTE are indeed connected through scattering theory. In this section, we will show an equivalent of the global solution 2.3 using the radiative transfer equation. To achieve this, we define the operator $\mathcal{S}^1$:

$$\mathcal{S}^1 = \int_0^r \sigma_s(\mathbf{x}', \vec{\omega}) \int_{4\pi} [\ ] p(\mathbf{x}', \vec{\omega}', \vec{\omega})(\vec{n} \cdot \vec{\omega}') d\omega' T_{r-r'}(\mathbf{x}', \vec{\omega}) dr'.$$

Note on how this operator is similar to the second term of Equation 2.13. Now, let us consider a bounded medium, with a starting radiance distribution on the surface $L_0(\mathbf{x}_o, \vec{\omega}_o)$ at a boundary point $\mathbf{x}_o$, where $\vec{\omega}_o$ points towards the inside of the medium. We can extend this initial radiance to any point $\mathbf{x} = \mathbf{x}_o + r\vec{\omega}_o$ of the medium:

$$L^0(\mathbf{x}, \vec{\omega}_o) = L^0(\mathbf{x}_o, \vec{\omega}_o) T_r(\mathbf{x}_o, \vec{\omega}_o).$$

After defining L^0, we can define an n-ary radiance function L^{n+1} recursively using operator $\mathcal{S}^1$:

$$L^{n+1} = L^n \mathcal{S}^1.$$

The n-ary radiance can easily be calculated using a continuous application of the S operator. If we define a new operator $\mathcal{S}^{n+1} = \mathcal{S}^1 \mathcal{S}^n$, one can show in a straightforward way that

$$L^n = L^0 \mathcal{S}^n$$

for every scattering order n. We can bring this process to infinity by defining the two quantities:

$$L = \sum_{j=0}^{\infty} L^j, \qquad \mathcal{S} = \sum_{j=0}^{\infty} \mathcal{S}^j.$$

In this particular case, we define $\mathcal{S}^0 = \mathcal{I}$, where $\mathcal{I}$, is the identity operator for which $f\mathcal{I} = f$ for every choice of f. That leads to the formulation:

$$L = \sum_{j=0}^{\infty} L^j = \sum_{j=0}^{\infty} L^0 \mathcal{S}^j = L^0 \mathcal{S}. \tag{2.14}$$

It is simple to prove that this formulation satisfies the integral form of the radiative transfer equation:

$$\begin{aligned} L &= L^0 \mathcal{S} \\ &= L^0 \left(\mathcal{I} + \mathcal{S}^1 + \sum_{j=2}^{\infty} \mathcal{S}^j \right) \end{aligned}$$

$$
\begin{aligned}
&= L^0 \left(\mathcal{I} + \mathcal{S}^1 + \sum_{l=1}^{\infty} \mathcal{S}^{l+1} \right) \\
&= L^0 \left(\mathcal{I} + \left(\mathcal{I} + \sum_{l=1}^{\infty} \mathcal{S}^{l} \right) \mathcal{S}^1 \right) \\
&= L^0 \left(\mathcal{I} + \mathcal{S}\mathcal{S}^1 \right) \\
&= L^0 + (L^0\mathcal{S})\mathcal{S}^1 \\
L &= L^0 + L\mathcal{S}^1,
\end{aligned}
$$

where the last equation corresponds to the integral form of the radiative transfer equation in Equation 2.13.

Note that, since the solution of the radiance must be unique (proof is omitted), the above calculated L in functional form corresponds to the radiance L calculated from Equation 2.3. This tells us that calculating the outgoing radiance through the BSSRDF is the same as integrating the radiance across all the possible paths scattering within the material. This fundamental connection justifies comparing BSSRDF renderings, an approximation of equation 2.3, with volume path tracing, in itself an approximation of Equation 2.14.

2.2.5 Rendering scattering media

Now, we want to use our formulas to derive a way to render scattering materials. The usual technique used in rendering to solve the rendering equation is Monte Carlo integration with importance sampling [Kalos and Whitlock, 2008]. Let us assume we need to solve the integral

$$I = \int_a^b f(x)dx.$$

If we draw samples uniformly samples in $[a, b[$, we can define a N-th estimator for I:

$$\hat{I}_N = \frac{b-a}{N} \sum_{i=1}^{N} f(x_i).$$

From the law of large numbers, $\hat{I}_N \to I$ for $N \to \infty$. To improve the convergence rate, we use importance sampling. If we know a distribution $\mathrm{pdf}(x)$ that is somewhat close in shape to $f(x)$ and from which it is easy to draw samples $x_i \in [a, b]$. We can rewrite the integral I as :

$$I = \int_a^b \frac{f(x)}{\mathrm{pdf}(x)} \mathrm{pdf}(x)dx,$$

which we can evaluate again using Monte Carlo as:

$$\hat{I}_N = \frac{b-a}{N} \sum_{i=1}^{N} \frac{f(x_i)}{\text{pdf}(x_i)}.$$

Where the x_i follow the distribution $\text{pdf}(x)$. As before, $\hat{I}_N \to I$ for $N \to \infty$.

2.2.5.1 Volume path tracing

Now let us apply importance sampling to obtain a first technique to render scattering materials, namely *volume path tracing* [Rushmeier, 1988]. Volume path tracing is the classical formulation to render participating media, and it is usually used to generate reference images. In our Contribution I, where we strive for comparing images with photographs, we use a GPU-optimized version of volume path tracing to efficiently render glasses of apple juice.

We start by using the integral form of the radiative transfer equation 2.13. For the sake of this example, we assume a non emissive homogeneous medium, so that the coefficients do not depend on position and direction within the medium. We also assume index matched media, so we can simplify the interaction at the boundary.

The algorithm, traces a ray from the camera direction $-\vec{\omega}$. The ray hits the surface at some point $\mathbf{x}_o$. We need to evaluate $L_r(\mathbf{x}_o, \vec{\omega})$. We first evaluate r as the distance to the other side of the medium, hitting a point $\mathbf{x}_t = \mathbf{x}_o - r\vec{\omega}$. Because of the assumption of homogeneous medium, the transmittance is easily evaluated:

$$T_r(\mathbf{x}_t, \vec{\omega}) = \exp\left(-\int_0^r \sigma_t dr'\right) = e^{-\sigma_t r}.$$

We evaluate $L^0(\mathbf{x}_t, \vec{\omega})$ by continuing the path outside the medium in direction $-\vec{\omega}$. Now, we want to evaluate the second part of Equation 2.13, which is $L_r^*(\mathbf{x}_o, \vec{\omega})$ from Equation 2.10. For this, we use Monte Carlo integration with importance sampling. We first reparameterize the integral by imposing $s = r - r'$:

$$L_r^*(\mathbf{x}_o, \vec{\omega}) = \int_0^r \sigma_s \int_{4\pi} L(\mathbf{x}', \vec{\omega}') p(\mathbf{x}', \vec{\omega}', \vec{\omega}) d\omega' T_s(\mathbf{x}', -\vec{\omega}) ds.$$

Note that $\mathbf{x}' = \mathbf{x}_o - s\vec{\omega} = \mathbf{x}_t + r'\vec{\omega}$. Now we can write the Monte Carlo estimator sampling both integrals:

$$\hat{L}_r^*(\mathbf{x}_o, \vec{\omega}) = \sum_{p=1}^{N} \sum_{q=1}^{M} \frac{\sigma_s L(\mathbf{x}_o - s_p, \vec{\omega}, \vec{\omega}_q') p(\mathbf{x}_o - s_p, \vec{\omega}, \vec{\omega}_q', \vec{\omega}) \exp(-\sigma_t s_p)}{\text{pdf}(s_p)\text{pdf}(\vec{\omega}_q')}. \quad (2.15)$$

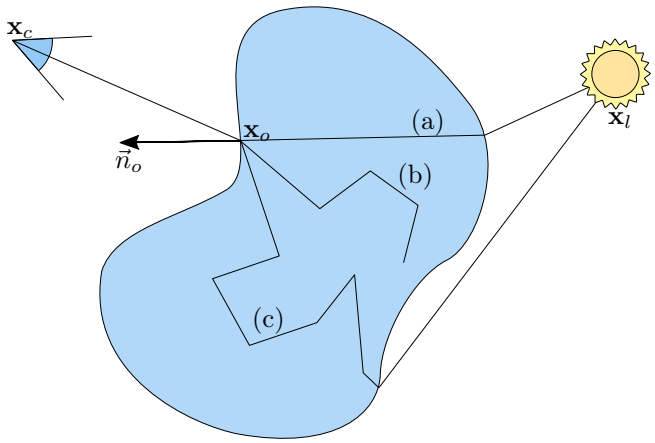


Figure 2.7: Volumetric path tracing, with various paths going through the medium from camera $\mathbf{x}_c$ to light $\mathbf{x}_l$. Path (a) represents direct transmission to calculate $L^0(\mathbf{x}_t, \vec{\omega})$. Path (b) represents a path killed through absorption, while path (c) successfully exits the volume and connects to the light.

For now, we assume that $s_p < r$, i.e. we are always sampling a point within the medium. Now, we need to choose the pdfs. We can sample s_p according to the formula:

$$s_p = \frac{-\ln(1-\xi)}{\sigma_t},$$

for $\xi \in [0, 1)$, that corresponds to a $\text{pdf}(s_p) = \sigma_t \exp(-s_p \sigma_t)$. Once we have chosen s_p, we can move onto sampling $\vec{\omega}'_q$. For standard phase functions such as Henyey-Greenstein [Henyey and Greenstein, 1940], it is possible to analytically find a way to importance sample a vector $\vec{\omega}'_q$. So, if the vector is carefully chosen, we have $\text{pdf}(\vec{\omega}'_p) = p(\mathbf{x}_o - s_p\vec{\omega}, \vec{\omega}'_q, \vec{\omega})$.

Plugging it in 2.15 we get the form

$$\hat{L}^*_r(\mathbf{x}_o, \vec{\omega}) = \sum_{p=1}^{N} \sum_{q=1}^{M} \alpha L(\mathbf{x}_o - s_p, \vec{\omega}, \vec{\omega}'_q),$$

where $\alpha = \sigma_s/\sigma_t = \sigma_s/(\sigma_s + \sigma_a)$ is called the *single scattering albedo*. Note that $0 < \alpha < 1$. Finally, we can introduce an additional importance sampling. We introduce a scattering probability $\text{pdf}_s = \alpha$. We decide then if we want to either scatter of absorb the photon, using Russian roulette [Arvo and Kirk, 1990]. If we decide to absorb the photon, we simply terminate the path. With

this procedure in place, we get our final formulation

$$\hat{L}_r^*(\mathbf{x}_o, \vec{\omega}) = \sum_{p=1}^{N}\sum_{q=1}^{M} \alpha \frac{L(\mathbf{x}_o - s_p, \vec{\omega}_q')}{\mathrm{pdf}_s} = \sum_{p=1}^{N}\sum_{q=1}^{M} L(\mathbf{x}_o - s_p, \vec{\omega}_q').$$

From this result, we see that we just need to recursively evaluate radiance to get the final result. At the next recursive step, we will evaluate a new s_p and $\vec{\omega}_q'$, continue the process with probability α and so on. This creates a random walk within the medium, that does not terminate for $s_p < r$. If $s_p > r$, it means that we are exiting the medium. In this case, we continue path tracing from the exit point in the last evaluated direction $\vec{\omega}_q'$.

2.2.5.2 The extended rendering equation

Instead of using the various forms of the radiative transfer equation, we can use the form of the rendering equation in Equation 2.3. Jensen and Buhler [Jensen and Buhler, 2002] proposed one such technique, namely hierarchical integration. In general, we want to be able to sample points and directions on the surface of the scattering medium, so that we can solve the rendering equation via Monte Carlo integration using

$$\hat{L}(\mathbf{x}_o, \vec{\omega}_o) \approx \sum_{p=1}^{N}\sum_{q=1}^{M} \frac{S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})(\vec{n}_{i,p} \cdot \vec{\omega}_{i,q})}{\mathrm{pdf}(\vec{\omega}_{i,q})\mathrm{pdf}(\mathbf{x}_{i,p})}.$$

This formulation needs that a BSSRDF, either measured or analytical, is available. We refer to our note in Appendix VII for a rendering technique that uses this estimator for the extended rendering equation, that we use in Contribution III as an unbiased reference. Also, the technique from [Mertens et al., 2003] makes use of this approximation, which we derived again in Appendix IX.

2.2.6 Analytical BSSRDF Models

So far, we have defined the BSSRDF as a proportionality constant between radiometric quantities. Given some assumptions about light propagation in materials, it is possible to derive analytical formulas for a BSSRDF for practical use in rendering. In the case of analytical models, Jensen et al. [2001] first provided an analytical BSSRDF function based on the dipole solution in the work of Farrell et al. [Farrell et al., 1992] (the standard dipole). In this section, we outline the derivation of three analytical models, the standard dipole [Jensen et al., 2001], the better dipole [d'Eon, 2012] and the directional dipole [Frisvad

et al., 2014]. Our note in Appendix VI provides additional derivation details for the interested reader. Note that more advanced dipoles exist, in particular the fully directional forward dipole [Frederickx and Dutré, 2017]. We refer to the original paper for details on the derivation, which is out of the scope of this thesis.

Generically, the light exiting from a scattering medium at a point $\mathbf{x}_o$ at a direction $\vec{\omega}_o$ can be thought as coming from three contributions: direct transmission, single and multiple scattering. If we define $\vec{\omega}_{21}$ as the refraction of $\vec{\omega}_o$ into the medium oriented towards the surface, direct transmission is the contribution of the light coming from direction $-\vec{\omega}_{21}$ that does not scatter along the path (see Equation 2.8). The single scattering term comes from light that enters at any point on the surface, scatters once, then aligns with $\vec{\omega}_{21}$ to refract out in direction $\vec{\omega}_o$. The multiple scattering term is similar, but it includes light that has scattered more than once before exiting the medium.

Single and multiple scattering are usually separated due to their different characteristics. Single scattering is highly peaked and directional, while multiple scattering can be handled as a stochastic process. The three analytical models we describe model the scattering of light as a diffusion process [Stam, 1995]. As such, the standard and better dipole model multiple scattering only, while the directional dipole models also part of the single scattering contribution.

As before, we define operators to simplify notation:

$$\mathcal{G} = \int_{4\pi} [\ ] d\omega, \quad \vec{\mathcal{G}} = \int_{4\pi} [\ ] \vec{\omega} d\omega.$$

We now use spherical harmonics to simplify radiance [Case and Zweifel, 1967]:

$$L(\mathbf{x}, \vec{\omega}) = \frac{1}{4\pi}\phi(\mathbf{x}) + \frac{3}{4\pi}\vec{\omega} \cdot \mathbf{E}(\mathbf{x}),$$

where $\phi = L\mathcal{G}$ and $\mathbf{E} = L\vec{\mathcal{G}}$. Once we combine the radiative transfer equation and the diffusion approximation, we get the form:

$$(D\nabla^2 - \sigma_a)\phi(\mathbf{x}) = -Q_0(\mathbf{x}) + 3D\nabla \cdot \mathbf{Q}_1(\mathbf{x}, \vec{\omega}), \tag{2.16}$$

where $Q_0 = q\mathcal{G}$ and $\mathbf{Q}_1 = q\vec{\mathcal{G}}$ are the integrals of the emission term in the radiative transfer equation $q(\mathbf{x}, \vec{\omega})$ (see Equation 2.1). The choice of q, or *source term*, leads to different approximations. Standard and better dipole use a point source term, while the directional dipole uses a ray source. Equation 2.16 is a particular case of screened Poisson equation, that can be explicitly integrated [Fetter et al., 2003]. For a point source term, we obtain:

$$\phi(\mathbf{x}) = \frac{\alpha' \Phi_i}{4\pi D} \frac{e^{-\sigma_{\mathrm{tr}} r}}{r}. \tag{2.17}$$

And for the ray source term, we get:

$$\phi(\mathbf{x}) = \frac{\Phi_i}{4\pi D}\frac{e^{-\sigma_{\mathrm{tr}} r}}{r}\left(1 + 3D\frac{1+\sigma_{\mathrm{tr}} r}{r}\cos\theta\right). \tag{2.18}$$

Now that we have a formulation for the fluence, we need to connect it to the BSSRDF. First, we consider only the part of the BSSRDF depending on the diffusion approximation S_d. From the definition of BSSRDF in Equation 2.2 and assuming a pure Fresnel boundary we get to this formulation in terms of the diffusive radiant exitance M_d

$$S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o) = \frac{1}{4\pi T_{12} C_\phi(1/\eta)}\frac{dM_d(\mathbf{x}_o)}{d\Phi_i(\mathbf{x}_i, \vec{\omega}_i)}.$$

Through the diffusion approximation, we can approximate the diffusive radiant exitance M_d as

$$M_d(\mathbf{x}_o) = C_\phi(\eta)\phi(\mathbf{x}_o) - C_{\mathbf{E}}(\eta) D \nabla\phi(\mathbf{x}_o)\cdot\vec{n}_o,$$

where T_{12} is the incoming Fresnel coefficient, C_ϕ and $C_{\mathbf{E}}$ are the first two orders of Fresnel integrals, which can be approximated by analytical formulas [d'Eon and Irving, 2011]. The value η is the index of refraction of the medium. By plugging in our solutions for the fluence in Equations 2.17 and 2.18, we get the analytical formula for a BSSRDF if we disregard the fact that light does not scatter outside the medium. In the case of a point source term, we obtain

$$S_d(\mathbf{x}_i, \mathbf{x}_o) = \frac{1}{4C_\phi(1/\eta)}\frac{\alpha'}{4\pi^2}\frac{e^{-\sigma_{\mathrm{tr}} r}}{r^3}\left[C_\phi(\eta)\frac{r^2}{D} + C_{\mathbf{E}}(\eta)(1+\sigma_{\mathrm{tr}} r)\mathbf{x}\cdot\vec{n}_o\right],$$

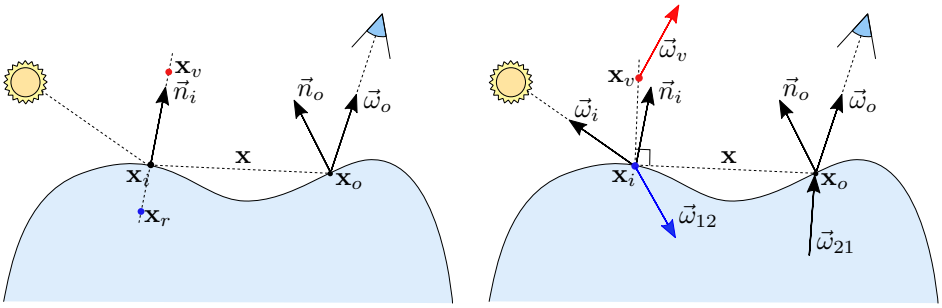
where $\mathbf{x} = \mathbf{x}_o - \mathbf{x}_i$ and $r = \|x\|$. This is the term used for the better dipole [d'Eon, 2012]. We can then further simplify the obtained monopole by disregarding Fresnel effects ($\eta = 1$), obtaining $C_\phi(\eta) = 0$, $C_{\mathbf{E}}(\eta) = 1$ and $C_\phi(1/\eta) = 1/4$. This leads to

$$S_d(\mathbf{x}_i, \mathbf{x}_o) = \frac{\alpha'}{4\pi^2}\frac{e^{-\sigma_{\mathrm{tr}} r}}{r^3}(1+\sigma_{\mathrm{tr}} r)\mathbf{x}\cdot\vec{n}_o,$$

which is used for the standard dipole [Jensen et al., 2001]. Finally, if we use the ray source solution, we obtain

$$S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o) = \frac{1}{4C_\phi(1/\eta)}\frac{1}{4\pi^2}\frac{e^{-\sigma_{\mathrm{tr}} r}}{r^3}\left[C_\phi(\eta)\left(\frac{r^2}{D} + 3(1+\sigma_{\mathrm{tr}} r)\mathbf{x}\cdot\vec{\omega}_{12}\right)\right.$$
$$\left. - C_{\mathbf{E}}(\eta)\left[3D(1+\sigma_{\mathrm{tr}} r)\vec{\omega}_{12}\cdot\vec{n}_o - \left((1+\sigma_{\mathrm{tr}} r) + 3D\frac{3(1+\sigma_{\mathrm{tr}} r) + (\sigma_{\mathrm{tr}} r)^2}{r^2}\mathbf{x}\cdot\vec{\omega}_{12}\right)\mathbf{x}\cdot\vec{n}_o\right]\right],$$

which is used for the directional dipole [Frisvad et al., 2014]. As we mentioned,



Standard and better dipole Directional dipole

Figure 2.8: Placement of virtual (red) and real (blue) sources for the dipoles described in the text.

all these solutions are suitable only when light is allowed to scatter within an infinite medium with coefficients σ_s, σ_a and g. To get a full BSSRDF, we need to introduce a boundary. We assume a refractive boundary with relative index of refraction η as the ratio of the index of refraction of the incoming medium to the index of the refraction of the scattering medium. For the vast majority of analytical BSSRDFs, we derive a half-space solution based on a plane surface, then correct the formulas for non planar geometry.

To achieve a solution for a point $\mathbf{x}_o$ on a semi infinite plane with normal $\vec{n}_o$, we let the fluence ϕ vanish at the boundary. This leads for the following boundary condition for the diffusion approximation:

$$\phi(\mathbf{x}_o) - 2Ad_e(\vec{n}_o \cdot \nabla)\phi(\mathbf{x}_o) = 0$$

Where d_e is called extrapolation distance, $A(\eta)$ is a term that accounts for mismatched index of refraction at the boundaries. The idea here is that the diffusion approximation matches better the exact solution if we let the fluence vanish not at the boundary, but at an *extrapolated boundary* at distance $2Ad_e$ along the normal direction. This boundary condition can be satisfied by introducing two light sources, a positive *real* source and a negative *virtual* source, that together form a *dipole configuration*. Standard and better dipole use two point lights in the configuration to the left of Figure 2.8, with the real source (blue) below and the virtual source (red) above the surface. The directional dipole uses a real ray source on $\mathbf{x}_i$, and a virtual ray source above the surface (see right of Figure 2.8). The solutions for infinite media reported above are then evaluated for the real and the virtual source, and the results are then combined to obtain the final BSSRDF value for rendering.

As we have seen, the complexity of the dipole greatly changes depending on how good the approximation is. The standard dipole, given its pure dependency

on r, can be approximated quite efficiently, especially as a series of Gaussian convolutions. The drawback of these dipole is that directional effects are lost, giving often an overly blurred result. In our contribution III we will see how do we manage to create a interactive rendering technique that can handle the directional effects introduced by the directional dipole.

2.3 Non-participating media

In this section, we will introduce some simplifications to the scattering theory introduced in Section 2.2. We introduce the description of two new type of media, transparent and opaque. Moreover, we introduce the BRDF, a reflectance function far more used in real-time applications than the BSSRDF.

2.3.1 Transparent media

Multiple of our contributions make use of glass, in particular Contributions I and II. In these contributions, we describe glass as a non-scattering medium, i.e. where $\sigma_s = 0$. This greatly simplifies the equations involved, leaving only the direct transmission term L_r^0 in Equation 2.10. This implies that the only noticeable attenuation comes from absorption, since $\sigma_t = \sigma_a$. Moreover, glass objects usually have a specular interface defined by a relative index of refraction η. This is a good approximation for clear polished glass as the one we render in Contribution II. We discuss in contribution I how to include a diffuse term in the interface to model tiny scratches on the surface. We can easily render glass in a path tracing environment using the technique in in Section 2.2.5.1, leaving out the random walk and including the direct transmission contribution.

2.3.2 Opaque media

2.3.2.1 The BRDF

The methods that include scattering media we described in Section 2.2.5 are usually quite complex to render, requiring a high number of samples to converge. For many materials such as plastic or metal, the scattering process is quite restricted around the point of emergence $\mathbf{x}_o$. For these particular materials, we can introduce some simplifications to obtain more tractable functions. We assume that the BSSRDF is limited across a small area around the emergence

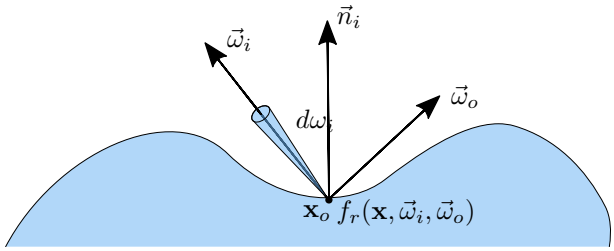


Figure 2.9: Configuration in which we define the BRDF for a generic surface.

point $\mathbf{x}_o$, and zero everywhere else. In this configuration, we can assume that the radiance is constant across the plane ($L_i(\mathbf{x}_i, \vec{\omega}_i) \approx L_i(\vec{\omega}_i)$). We also need to assume that the material is locally isotropic, i.e. its properties do not change across the surface. In this case, we can approximate the outgoing radiance as:

$$dL(\mathbf{x}_o, \vec{\omega}_o) \approx \int_A dL(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = \int_A S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) d\Phi_i(\mathbf{x}_i, \vec{\omega}_i).$$

By the definitions of flux and irradiance and the assumption of constant radiance outlined above, we obtain

$$\begin{aligned} dL(\mathbf{x}_o, \vec{\omega}_o) &= \int_A S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_i, \vec{\omega}_i)(\vec{n} \cdot \vec{\omega}_i) dA_i d\omega_i \\ &\approx L_i(\vec{\omega}_i)(\vec{n} \cdot \vec{\omega}_i) d\omega_i \int_A S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) dA_i \\ &= dE_i(\vec{\omega}_i) \int_A S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) dA_i. \end{aligned}$$

The last integral eliminates the dependency on $\mathbf{x}_i$, as it is a function purely dependent on the point of emergence $\mathbf{x}_o$ the two angular vectors $\vec{\omega}_i$ and $\vec{\omega}_o$. This proportionality constant between incoming irradiance and outgoing radiance f_r

$$dL(\mathbf{x}_o, \vec{\omega}_o) = f_r(\mathbf{x}_o, \vec{\omega}_i, \vec{\omega}_o) dE_i(\mathbf{x}_o, \vec{\omega}_i), \tag{2.19}$$

is called the *bidirectional reflectance distribution function* (BRDF) [Nicodemus et al., 1977]. The BRDF is measured in $[\mathrm{sr}^{-1}]$. As before, we can obtain the reflected radiance from the definition above:

$$L(\mathbf{x}, \vec{\omega}) = \int_\Omega f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) L_i(\mathbf{x}, \vec{\omega}_i)(\vec{n} \cdot \vec{\omega}_i) d\omega_i \quad [\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]. \tag{2.20}$$

This is called the reflected radiance equation. As before, we can add the emitted radiance L_e to obtain the traditional form of the rendering equation [Kajiya, 1986]. We can easily extend the two properties of the BSSRDF in Equations 2.4

and 2.5. Conservation of energy imposes

$$0 \leq \int_{\Omega} f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o)(\vec{n} \cdot \vec{\omega}_i) d\omega_i \leq 1.$$

Helmholtz reciprocity translates to

$$f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) = f_r(\mathbf{x}, \vec{\omega}_o, \vec{\omega}_i).$$

From the reflectance equation above, we can see why BRDFs are usually employed in real-time applications, since BRDF require one less integral to solve (compare Equations 2.3 and 2.20). We then use BRDFs in our Contributions II, IV and V where the focus is not specific to scattering media. In applications such as games we may have a finite number of point and directional lights, which simplifies the integral in Equation 2.20 to a sum over all the lights.

2.3.2.2 Empirical BRDFs

Given the low dimensionality of the BRDF, it is possible to actually measure and tabulate it, so that it can be used in future renderings. We use tabulated BRDFs multiple times in our Contributions II and V, given that they allow us to easily achieve a photorealistic look. If we assume a non-spatially varying BRDF (dropping the dependency from $\mathbf{x}_o$ in Equation 2.19) and expressing each vector $\vec{\omega}$ in terms of its polar angles θ, ϕ, we obtain a four dimensional function:

$$f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) = f_r(\theta_i, \phi_i, \theta_o, \phi_o).$$

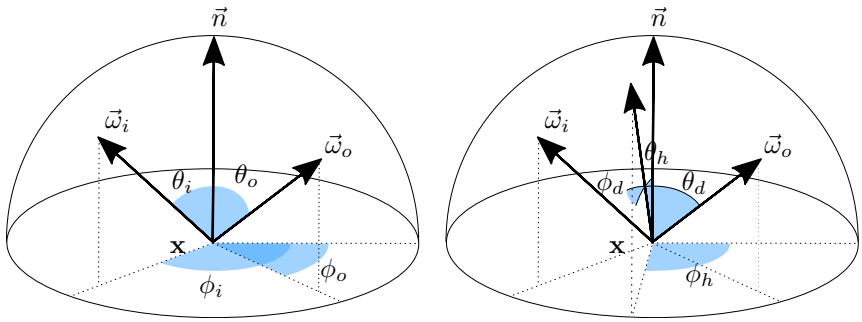
Moreover, for empirical BRDFs we usually assume the BRDF to be *isotropic*, so that it depends only on the relative angle $\phi_{\text{diff}} = \phi_o - \phi_i$:

$$f_r^{\text{iso}}(\theta_i, \phi_i, \theta_o, \phi_o) = f_r^{\text{iso}}(\theta_i, \theta_o, \phi_{\text{diff}}).$$

Anisotropic materials (e.g. brushed metal) are rarer in nature and are usually created using some sort of manufacturing process. This naïve representation, presented in Figure 2.10, can be used to store BRDFs in a compact way, though with some issues. Generally, BRDFs tend to have a highly peaked region around the half vector:

$$\vec{\omega}_h = \frac{\vec{\omega}_i + \vec{\omega}_o}{\|\vec{\omega}_i + \vec{\omega}_o\|},$$

where there is usually a strong peak due to reflection, or *specular highlight*. If we use the naïve parametrization, we would encounter some discretization issues around $\vec{\omega}_h$. To avoid this loss of precision, it is important to represent the BRDF more accurately around the half vector. So for measured datasets such as the MERL dataset [Matusik et al., 2003], the Rusinkiewicz half-difference



Polar representation
$f_r(\theta_i, \phi_i, \theta_o, \phi_o)$

Rusinkiewicz representation
$f_r(\theta_h, \phi_h, \theta_d, \phi_d)$

Figure 2.10: Different representations for BRDFs. To the left, standard representation using polar coordinates. To the right, the representation from [Rusinkiewicz, 1998].

parametrization is used (see Figure 2.10). This representation measures angles relative to the half vector instead of the surface normal:

$$f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) = f_r(\theta_h, \phi_h, \theta_d, \phi_d),$$

where θ_h, ϕ_h are the spherical coordinates of $\vec{\omega}_h$, and where θ_d, ϕ_d are the spherical coordinates of $\vec{\omega}_i$ rotated so that $\vec{\omega}_h$ matches the reference frame of $\vec{n}$. In this case, isotropic materials can be described by dropping the dependency on ϕ_d, i.e. the overall rotation of the system around the normal.

After we have chosen our representation, we can pack our isotropic BRDF in a 3D data structure. Note that similar packing representations exist also for BSSRDFs [Donner et al., 2009], but the high dimensionality of the BSSRDF (5 up to 14 dimensions) renders these representations somewhat less practical. See the note in Appendix VIII for a derivation on how to compute a simulated empirical BSSRDF.

2.4 Rendering techniques

In the previous sections, we introduced theory and algorithms for rendering physically based materials, without going into details on how the different algorithms are implemented. To conclude our background section, we introduce two of the main paradigms used to render physically based materials on GPUs, namely rasterization and ray tracing. In our contribution, we use both techni-

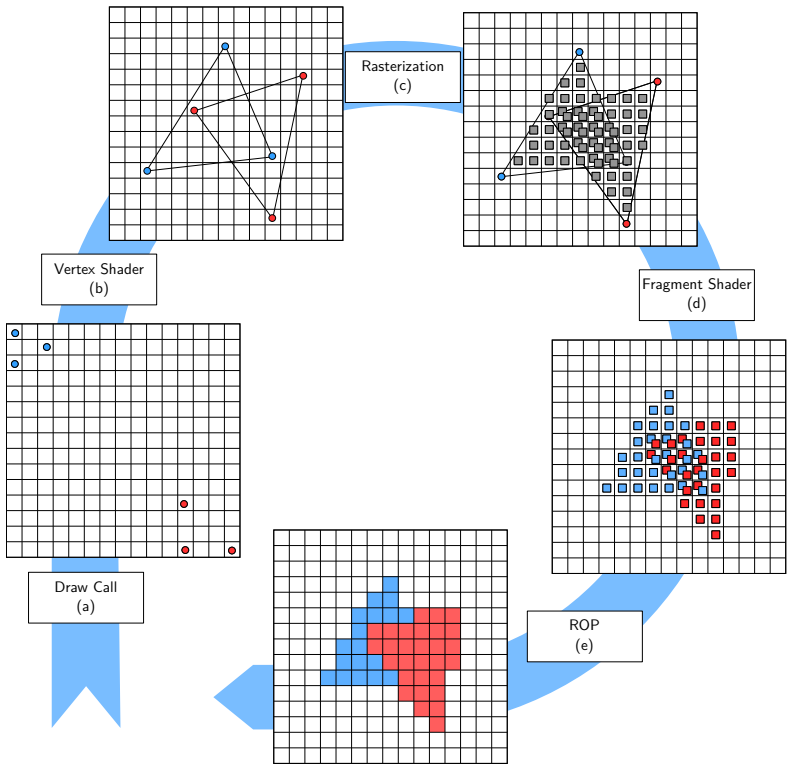


Figure 2.11: The rasterization pipeline. Following the arrow: bottom left, vertices in local coordinates are fetched from memory; top left, vertices are disposed by the vertex shader, then the triangle is assembled; top right, fragments (gray) are generated; bottom right, fragments are shaded; bottom, fragments are combined by a depth test to form the final image.

ques (rasterization in Contributions III and V, ray tracing in Contributions II, I and IV), choosing either one depending on the application constraints.

2.4.1 Rasterization

The first of the two techniques is rasterization. In this technique, some rendering primitives, triangles, are scanned one by one and then drawn on the screen. One example of algorithms used to transform triangles into pixel-size elements (also called *fragments*) is scanline rendering [Wylie et al., 1967], though

it depends on the GPU vendor. By its nature, rasterization is highly parallelizable, since every primitive can be drawn in parallel. On modern graphics cards, the rasterization process is performed in hardware as part of the *graphics pipeline*, a highly fine tuned sequence of steps, both hardware and software, to render triangles efficiently. A simplified version of the pipeline is illustrated in Figure 2.11. The pipeline is composed of programmable parts (called *shaders*) and hardware parts, that are only controllable through state flags. The core rasterization process, transforming primitives into fragments, is executed in parallel by a specified hardware unit.

Let us describe the life of a single triangle through the pipeline. Once the CPU issues the command to draw some triangles, a *draw call* (Figure 2.11, a), the pipeline starts on the GPU. For each of the three vertices composing a triangle, we independently execute a programmable part, called the *vertex shader*, that has to output a normalize screen space position. This stage allows operation such as model and perspective transformation (Figure 2.11, b). Once the vertices have been processed, the triangle primitive is assembled, then processed by the hardware rasterizer, generating a certain number of fragments (Figure 2.11, c). For each fragment, we execute another programmable shader, the *fragment shader*. This stage needs to output the final color of the fragment, so light interaction is usually added at this stage (Figure 2.11, d). Multiple attributes can be passed in between vertex and fragment shader, which will be interpolated using the triangle's barycentric coordinates. Once the fragments are generated, they need to be stored on the image plane. Note that multiple fragments can land on the same pixel, and that the order of landing of the fragments is not defined, since generally the pipeline execution is asynchronous. So, modern GPUs use another hardware unit, the Render OutPut unit (ROP, Figure 2.11, e) to determine how to store the fragments in the final image. This unit usually can perform depth test (via a Z-buffer) or blending, allowing us to obtain a consistent result across invocations.

This is only a simplified view of the full graphics pipeline. The full pipeline allows tessellation (domain and hull shaders), geometry manipulation (geometry shaders) or writing back into a vertex stream (transform feedback). There are also shaders that are completely detached from the graphics pipeline (compute shaders), allowing to perform tasks in parallel on the GPU that are not directly related to a graphics task.

Rasterization is extensively used in game development and real-time applications, given its high speed and performance predictability. However, it is not particularly well suited for propagating light across a scene, making it difficult to achieve complex optical effects, which is why these effects often require ad-hoc solutions.

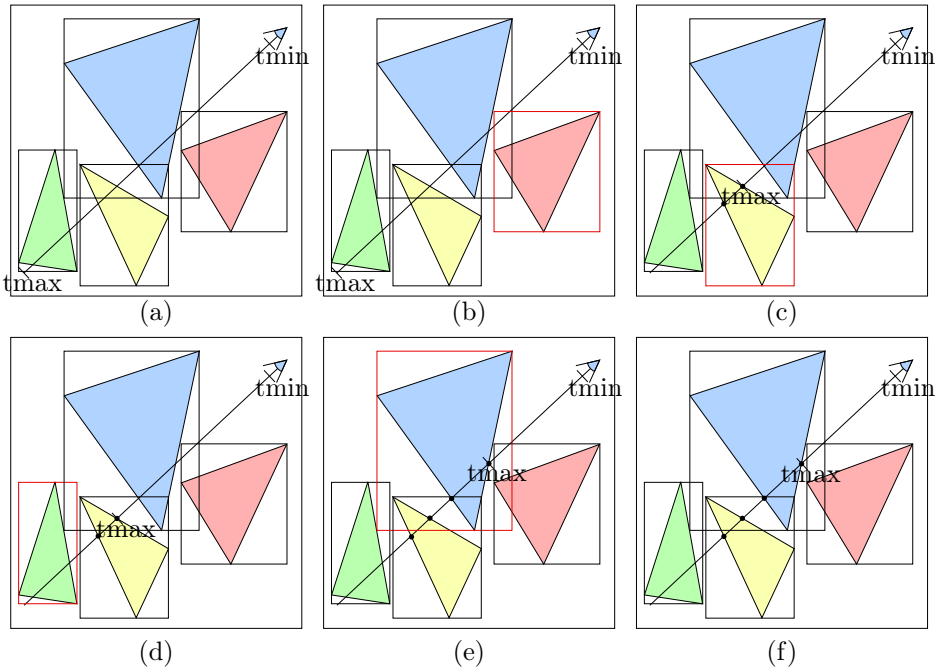


Figure 2.12: Ray tracing four triangles in two dimensions. The thicker line corresponds to the ray (a). At each step, we consider a different bounding box (thick red border). We illustrate different cases progressively: hitting bounding box without hitting primitive (b), hitting primitive (c), hitting bounding box but not hitting primitive because triangle is behind tmax (d), updating tmax to closest primitive hit (d). The last figure (f) shows all four any hit invocation (black dots) plus the closest hit (black dot with tmax).

2.4.2 Ray tracing

We start this section by describing *ray casting*, that can be considered the dual technique to rasterization. In rasterization, we match each triangle to its final position on the screen. In ray casting, we do the opposite: for each pixel on the screen, we find the corresponding triangles that land within that pixel. This equates to *shooting* a ray through the pixel, and intersecting it though the scene geometry. Once the hit point is found, attributes can be generated and the point shaded as in a fragment shader.

We note that, once the data structure is established, we can trace rays from any point to another in the scene. The term *ray tracing* describes this process of querying a data structure to generate intersection points. Depending on the algorithm used, ray tracing can be used as a base of multiple rendering algorithms, such as the volume path tracing we illustrated in Section 2.2.5.1. One of the first ray tracing algorithms was proposed by Appel [1968].

For the purpose of this thesis, we focus on GPU ray tracers. Modern GPU ray tracers build a tree data structure to efficiently lookup intersections. The most commonly used data structure is a *bounding volume hierarchy*, or BVH. In a BVH, each of the primitives (usually triangles, but custom primitives such as spheres, cubes or quadrilaterals are possible) is enclosed in a bounding box. The various bounding boxes are then arranged in a tree data structure for fast lookup. Once a ray is traced, it is first inexpensively tested against the bounding boxes. For the bounding boxes that are hit, the expensive primitive-ray intersection test is performed to check for a hit. Depending on the shading algorithm, a callback can be issued at every encountered intersection (*any hit*), or at the intersection closest to the ray origin (*closest hit*). The process to trace a single ray in a two dimensional example is illustrated in Figure 2.12.

Many challenges need to be solved to obtain an efficient GPU ray tracer. The first challenge is recursion. In the case of an algorithm such as path tracing, a probabilistic recursive ray tracing call is performed at the closest hits. So, an efficient ray tracer needs to keep track of the state at each intersection, generically through a recursion stack (OptiX [Parker et al., 2010] is such an example, where the stack is built automatically at the instruction level). The second challenge is on how to manage the data structure if the geometry of the material changes, through deformation or animation. Simple rigid transformations can be handled relatively inexpensively by the BVH, though more complicated operations need to be done if changes are performed at the primitive level. Modern ray tracing techniques such as the TRBVH [Karras and Aila, 2013] can rebuild part of the BVH on the fly without excessive memory consumption.

Given its state as a "natural" solution to light transport and its ability to model complex optical effects, ray tracing is commonly employed in the animated movie industry. Moreover, ray tracing scales better to scenes with a huge amount of triangles, such as in animated movies or virtual effects scenes. On the other hand, ray tracing is limited in the game developers community, due to smaller scenes and increased memory and lookup costs. Moreover, algorithms such as path tracing tend to give a noisier result, that is not acceptable in game applications.

Chapter 3

Related work

In this section, we present some related work on the overall topic of the thesis, namely physically based techniques brought into the interactive domain. We will make use of the background theory we presented in Chapter 2, to better discuss our contributions in Chapter 4. This section is not meant to be a complete survey of such techniques, but we will point to the most important contributions, referring to a number of individual papers and surveys for a more complete overview. In particular, we point to the survey by Ritschel et al. [2012] for an overview of various interactive techniques for global illumination. We also refer to the related work of the individual contributions in the appendices for more detailed literature in the context of each contribution.

As mentioned in Chapter 1, interactive rendering requires a compromise between accuracy and speed. There are various examples in literature on the compromises that are needed to improve this trade-off. Since the rendering time constraints are usually set, compromises are usually done in physical accuracy. We will start by discussing some of the assumptions that are usually employed in interactive rendering environments.

3.1 Typical approximations in physically based rendering

One of the natural approaches towards achieving interactive photorealistic rendering is making some assumptions about our physical simulation. In principle, we can describe any light interaction by running a brute-force simulation of all the photons interacting with the particles of the material, then measuring the number of photons arriving at a set light sensor. Of course, running this sort of computation, even for extremely simple scenes, in reasonable times is not possible. In general, Monte Carlo path tracing [Kajiya, 1986] with extension to scattering media [Rushmeier, 1988] is considered a ground truth technique to generate reference images. Path tracing is an unbiased technique (i.e. converges always to the correct solution), but it gives extremely noisy results, especially for particular configurations of light and camera positions. To solve this problem, the offline rendering community proposed various improvements to reduce variance in path tracing, such as importance sampling [Kirk and Arvo, 1991],multiple importance sampling [Veach and Guibas, 1995], Metropolis sampling [Veach and Guibas, 1995], Bidirectional path tracing [Veach and Guibas, 1997], manifold exploration [Jakob and Marschner, 2012], gradient-domain path tracing [Kettunen et al., 2015], etc. The book by Pharr et al. [2017] gives a good overview of different path tracing techniques. We described one of these importance sampling techniques in Section 2.2.5.1, where we derived a formulation for volume path tracing.

When we need interactivity, different approximations can be used. Once we know the underlying physical process, different choices can be made on a per material basis. For example, in the case of most metals, the subsurface scattering of light is very limited around the point of incidence, so the BRDF approximation described in Section 2.3.2.1 can be used. In the case of translucent materials, we can use the analytical BSSRDFs described in Section 2.2.6 instead of a fully path-traced simulation. Depending on the directionality of the scattering effects, different models can be used, at the price of increased rendering times. Moreover, for some particular materials, single scattering can be approximated by a BRDF, even for layered materials [Blinn, 1982; Hanrahan and Krueger, 1993]. In applications such as games or virtual reality, often it is not even possible to use BSSRDF models, using an approximate BRDF instead. This gives acceptable results, but since subsurface scattering effects are important, this results in a "waxy" effect, in particular for materials like skin.

Another approach, instead of simplifying the physical model, is to use real measured data. This is the approach for example used by the discretized BRDF described in Section 2.3.2.2. The drawback of measured materials is that they

often require large storage spaces. This is often a problem, since most rendering pipelines are already memory-bound. These tabulated models often have discretization issues, and require particular care in deciding a proper storage structure, such as the Rusinkiewicz parameterization for measured BRDFs [Rusinkiewicz, 1998]. Finally, measured data are limited to the setup used to generate them. Tabulation of the BSSRDF was attempted by Donner et al. [2009], where the BSSRDF is simulated as a Monte Carlo path traced simulation. Many assumptions are required to simulate the BSSRDF in reasonable times, reducing the fourteen dimensions of the BSSRDF to a more tractable five.

So far we have discussed material simplification. In general, the light hierarchy of a scene can also be object of simplification. Generically, a light is an object like any other in the scene, but it also emits light in the visible spectrum. In a path tracing context, lights are easily handled by the L_e term in Equation 2.2.2. However, since the number of lights is usually limited, and since the light position is generally known in advance, multiple techniques can be used to improve convergence [Shirley et al., 1996]. Further simplifications are possible: in games and real-time applications, lights often do not have an area extent. This is the case of directional, point and spot lights. This allows us to represent lights as delta functions and replace the integral over Ω^+ in Equations 2.20 and 2.3 with a sum over all the lights in the scene, with each contribution multiplied by a visibility factor. Recently, real-time polygonal lights have been introduced by Heitz et al. [2016].

We will briefly touch upon geometry and how it is represented in interactive applications. Various representations and primitives exist. However, in the vast majority of interactive applications triangular meshes are employed. This comes from the fact that rasterization and ray tracing algorithms can be greatly optimized by assuming a unique type of primitive. Triangles, compared to other primitives, have the advantage of being planar. Though GPUs are able to push more and more triangles due to improved hardware, excessive geometric detail still needs to be reduced in order to maintain acceptable frame rates. This is particularly true of massive scenes, where techniques as occlusion culling and level-of-detail [Clark, 1976] need to be employed to achieve interactive frame rates.

3.2 Rendering techniques

Once we have made our choice of physical model, as we discussed in the previous section, some other choices need to be done on the implementation side. These implementation choices account for the fact that our algorithm runs on

a discrete system with finite memory and processing power. In this thesis, we focus on GPU techniques, which thus usually exploit the massive parallelism offered by the streamed multiprocessing units on GPUs. In literature, we identified three common approaches employed by the various techniques, namely caching, precomputation and filtering. Each of these approaches leverages some approximations or introduces some limitations in order to work. In caching, we use some intermediate data structures to accelerate the per-frame rendering time. Caching also implies using data structure to efficiently reuse data, either spatially (to exploit cache coherence) or temporally, to amortize computation across frames. Precomputation, the second approach, allows to move some of the computation before the program actually executes, given some assumptions about the materials or geometry. Finally, filtering, the final approach, reconstructs missing information based on a sparse sampling of a target function. Note that techniques often fall into multiple approaches: each technique is usually a combination of caching, precomputation and filtering. Bearing this in mind, we will now proceed to present relevant theory for each approach.

3.2.1 Caching

In many cases in rendering, we need knowledge of both the local geometry around a geometric point (e.g. to estimate the occlusion of a point) in the scene and the overall global geometry (e.g. for global lighting effects). A number of techniques provide data structures to efficiently retrieve both the local and global geometry of a point. The most simple and widely used of these techniques is deferred shading [Saito and Takahashi, 1990], which rasterizes geometry, depth and positions into a highly optimized screen space data structure. This structure allows sampling of local geometry, and it can be used to implement various screen space techniques. Multiple G-buffers can be also be used to achieve global effects [Mara et al., 2016]. For global effects, a common approach is to create a traversal structure to efficiently implement ray tracing, such as the bounding volume hierarchies described in Section 2.4.2. Many optimized ray tracing techniques exploiting BVHs have been developed in recent years [Parker et al., 2010; Wald et al., 2014; Hendrich et al., 2017; Meister and Bittner, 2018]. We use these data structures through the interface exposed by the OptiX programming system [Parker et al., 2010] to implement efficient path tracing in out Contributions I and II (see Section 4.1). Once a traversal structure for ray tracing is in place, various classical ray tracing algorithms can be efficiently implemented, such as recursive ray tracing, path tracing, or volumetric path tracing. Davidovič et al. [Davidovič et al., 2014] provide a good survey about progressive path tracing techniques in a GPU context. Other data structures can be used, such as octrees [Glassner, 1988; Havran, 2000], or tracing rays directly in a screen space structure [Tanaka et al., 1986; McGuire and Mara, 2014;

Widmer et al., 2015].

Other techniques involve some form of light caching. One of the simplest light caching techniques, the render cache [Walter et al., 2002], can be done in screen space. This technique stores the radiance from a frame and reprojects it to the next, then filling holes created through visibility mismatches. Our Contribution IV on stable ray tracing is within this area of research (see Section 4.3). Screen space techniques can be used to efficiently render also scattering media [Nalbach et al., 2014], by solving the extended rendering equation 2.3 by sampling geometry in a local neighborhood. Other light caching techniques propagate illumination in the scene, store it, then render the scene again from the camera. This allows to achieve global effects. Photon mapping [Jensen, 1996] is a seminal paper in this regard. In this technique, photons are cast from light sources, bounced around the scene through ray tracing, then arranged into photon maps. Density estimation is then use to derive the final illumination from the maps. The overall efficiency can be enhanced by building a data structure for fast gathering of nearby photons. See the paper by Mara et al. [2013] for a fast GPU implementation. Instant radiosity [Keller, 1997] uses a similar approach, but uses gathering instead of density estimation in order to estimate the final radiance value at the exit point. These techniques are often named VPL (virtual point light) techniques, since each photon stored in the scene is treated as a small point light. Various literature deals with improving VPLs, including using the pixels in shadow map G-buffer as VPL sources, in the form of a reflective shadow map [Frisvad et al., 2005; Dachsbacher and Stamminger, 2005]. Other enhancements include VPL clustering [Walter et al., 2005; Bus et al., 2015], adding visibility [Ritschel et al., 2008], generalizing them into virtual area lights [Dong et al., 2009], or by enhancing them to store multiple views [Simon et al., 2015]. A comprehensive survey on VPL techniques is available from Dachsbacher et al. [2014]. Our Contribution III uses standard VPLs to transport outgoing scattered light (see Section 4.2).

Mixed techniques that combine geometry and light approximation are also possible. One example is radiosity [Goral et al., 1984], where we approximate the scene as a collection of geometric patches, then precompute the light transport in between patches. After this, the overall light transport problem can be described as solving a linear system. Another approach that employs both geometry simplification and light caching is point based global illumination [Bunnel, 2005; Christensen, 2008], where the scene is represented as a series of surface elements (surfels). In this technique, we first build a hierarchy of surfels. Surfels are then shaded. Finally, for each pixel in the final rendering, the relevant surfels falling within that pixel are rendered, obtaining the final result.

Other techniques use efficient data structures to do a volumetric light transport simulation. These techniques are particular suitable to render participating

media, but they can also be used to render traditional diffuse and glossy illumination. In the case of participating media, these techniques often use finite elements to solve the radiative transfer equation on a discretized grid [Fattal, 2009]. These particular techniques are used to obtain real-time results, such as in the case of light propagation volumes [Kaplanyan, 2009; Børlum et al., 2011], and further extended to discrete ordinate methods for point and directional lights [Elek et al., 2014]. In the case of diffuse and glossy illumination, various examples exist in the real-time rendering community. Grid-based radiance caches can be used [Nijasure et al., 2005], and also combined with ideas from reflective shadow maps to obtain radiance hints [Papaioannou, 2011; Vardis et al., 2014], that can handle multiple diffuse inter-reflections. Crassin et al. [2011] propose voxel cone tracing using an adaptive octree data structure to filter and propagate lighting. This can then evaluated with cone rays. Hoetzlein [2016] proposes and optimized GPU voxel based volumetric structure to visualize huge scientific datasets.

3.2.2 Pre-computation

In this section, we deal with the aspect of precomputation. By introducing some limitation in our scene, we can precompute some data for efficient rendering. Limitations include static or mostly static geometry, static lights, fixed materials or fixed cameras.

Relighting techniques [Nimeroff et al., 1994; Pellacini et al., 2005; Hašan et al., 2006] are a first example in which we require camera and geometry to be fixed. In this case, visibility and geometry are cached and reused across frames, allowing to interactively change lighting and materials. This is often necessary in a movie production pipeline, where due to artistic reasons lighting changes are more common than camera and geometry placement changes.

Precomputed radiance transfer [Sloan et al., 2002] assumes static geometry only (light and view can freely change). This family of techniques involve precomputing the radiance transfer at the surface for infinitely distant lights. Basis function are used to approximate lighting transfer at the surface, allowing slow preprocessing times but a fast evaluation via a dot product. The quality of the rendering depends on the number of basis functions used, though the memory and performance requirements then increase as well. We use this technique to approximate environment lighting in our Contribution V, as described in Section 4.4. Spherical harmonics are the most commonly used basis functions, but others such as Gaussians are possible [Green et al., 2006]. Precomputed radiance transfer does a good job representing low frequency lighting changes across the scene. The radiosity algorithm we described above is another algo-

rithm that precomputes light transport assuming static geometry.

If we assume static light and geometry, the whole incoming illumination at a point can be cached, then used to re-light objects moving through the scene. Moreover, indirect illumination can be precomputed via ray tracing and stored as additional texture maps, called lightmaps. This technique has been widely used in games, starting from John Carmack's Quake in 1996.

In scattering media, various precomputation are possible. We can also apply the precomputed radiance transfer technique, modified to work with translucent materials [Sloan et al., 2003]. Another approach is to precompute a grid, that can be used with a fast diffusion computation to render scattering in real-time [Wang et al., 2008]. Finally, some methods preprocess the existing mesh to create a multi-resolution mesh that can be used to propagate irradiance via finite elements, and this can handle deformable objects at interactive frame rates [Mertens et al., 2003; Li et al., 2013]. While taking a very different approach, such techniques are somewhat related to our Contribution III.

3.2.3 Filtering

The last branch of techniques we discuss is filtering. Filtering approaches regard reconstructing the final appearance based on a limited set of samples. As in previous sections, we start by describing screen space techniques. A big area of research includes anti-aliasing techniques, that involve improving appearance of undersampled features in the scene. Temporal anti-aliasing [Karis, 2014; Patney et al., 2016] involves recycling color information from the previous frame via motion vectors, and combining with the current color distribution, to achieve anti-aliasing across time. More recently, new techniques have been developed to filter noisy one sample Monte Carlo simulations to achieve a smooth temporally stable result. Work in this area includes pre-filtering [Crassin et al., 2015], advanced edge-aware bilateral filtering [Mara et al., 2017], temporal variance averaging [Schied et al., 2017], and machine-learning based filtering [Chaitanya et al., 2017]. The work from Mehta et al. [2013] proposes an adaptive filtering scheme to efficiently filter out high frequency Monte Carlo noise respecting physically based constraints. We also contribute to this body of work with our Contribution IV, described in Section 4.3.

Moving into more advanced structures, irradiance caching techniques [Ward et al., 1988; Tole et al., 2002] store the illumination at a limited set of pixels, then interpolating illumination across the points to achieve overall results. The technique can be further enhanced to work in volumes [Greger et al., 1998], and various heuristics in order to avoid light leaking though objects have been propo-

sed [Gautron, 2009]. Irradiance caching has been extended to radiance caching to include directional information [Křivánek et al., 2008; Scherzer et al., 2012]. Another set of techniques widely used in modern games is probe sampling [Levoy and Hanrahan, 1996; Hooker, 2016; McGuire et al., 2017; Silvennoinen and Lehtinen, 2017] where a spherical radiance map is stored at fixed points in the scene, called probes. The illumination is then interpolated across the probes to allow lighting of both static and dynamic geometries.

Screen space techniques can be effectively used also in the case of scattering media, approximating the scattering process as a series of Gaussian filters [d'Eon and Irving, 2011; Jimenez et al., 2015]. This effectively approximates the diffusion process using the standard or better dipole described in Section 2.2.6, leading to plausible results. The path integral formulation [Premože et al., 2003; Hegeman et al., 2005] and the narrow beam theory [Shinya et al., 2016] can be used instead of the diffusion approximation to propagate and filter the scattering contribution, giving interactive results.

As we have seen in this section, the space of interactive techniques that try to deliver a physically accurate result is huge. Most of these techniques rely on simplifications, assumptions and discretizations that allow fast rendering, often sacrificing physical accuracy in the process. In the next chapter, we will show how we improved upon current techniques to achieve a more physically accurate result.

Chapter 4

Contributions

In this section, after introducing relevant theory and related work, we finally discuss the different contributions created over the course of the Ph.D. studies. The goal of this section is to discuss the individual contributions in the light of the thesis goal. We refer to the text of the individual publications in Appendices I-V for the full details. Please note that most publications come with attached videos, that are linked at page xi.

From our discussion in Section 1.2 we argue that there is a need in the industry for photorealistic accurate interactive rendering, i.e. the top-left area of Figure 1.3. In many fields, people need immediate feedback on on the aspect of the final product. Some examples include visual inspection of produced parts, preview of 3D printed objects, artistic iterations for movie scenes, and prediction of the outcomes of an industrial process. We showed in Figure 1.5 where our contributions lie within the accuracy-time spectrum, addressing different accuracy and time targets depending on the application.

In this chapter, we will start by discussing the importance and the role of interactive photorealistic rendering presenting Contributions I and II. In the former, we make a case on why we need both fast and accurate rendering, in the form predicting the final appearance of cloudy apple juice from production parameters. In the latter Contribution II, we discuss how fast photorealistic rendering can be used to achieve accurate parameter estimation. We will then start di-

rendering photograph

Figure 4.1: Cloudy apple juice photographed and rendered the appearance model from Contribution I. In the model, we inferred apple particle concentration (0.8 g/l) and apple storage period (4 days) to match the photograph.

ving into the interactive techniques we contributed with. In Contribution III we discuss an interactive method to render with the directional BSSRDF described in Section 2.2.6. We will continue with Contribution IV, a caching and filtering technique leveraging interactive ray tracing to improve temporal stability of rendered images. Moving forward, we will describe in Contribution V an application of physically based rendering in a virtual reality environment. Finally, we will conclude by discussing some future directions we can expand our work.

4.1 Photorealistic rendering for appearance prediction and parameter estimation

We start our discussion from Contributions I and II. Our first discussion point is about validating physically based rendering. In literature, the emphasis is often onto creating rendering models and techniques that effectively approximate an arbitrary radiometric process. However, not much emphasis is put into validating the developed models with an image of a physical object, like we do

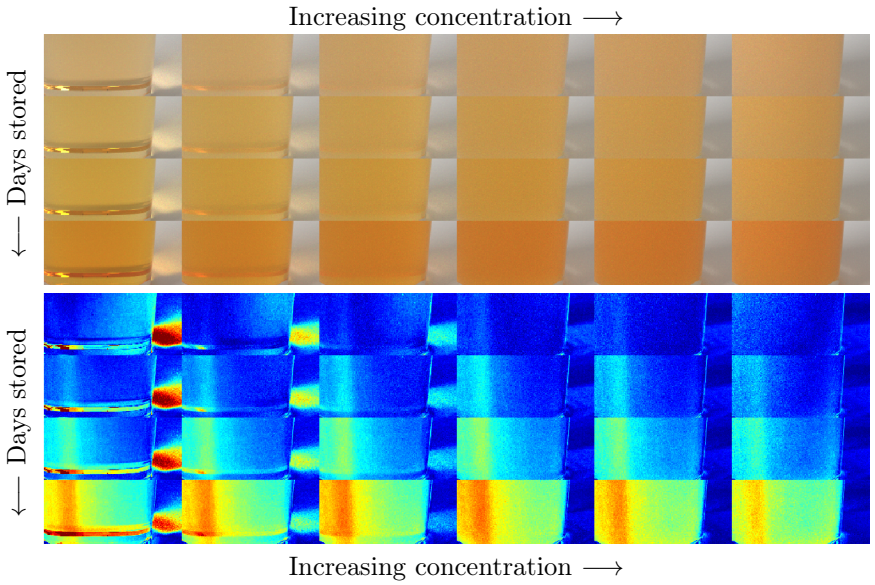


Figure 4.2: Comparing a patch of various renderings of cloudy apple juice in Contribution I. The horizontal axis shows increasing concentration, while the vertical axis shows the number of days the apples have been stored before pressing them. The top image shows the actual renderings, the bottom image shows false-color comparisons to reference (red is higher error).

in Figure 4.1 from Contribution I. In the figure, we compare two pictures of cloudy apple juice, one a picture of an actual glass of juice, one a rendering representation of it using interactive volumetric path tracing (12 samples per pixel per second), with the algorithm described in Section 2.2.5.1. In literature, authors usually rely on path traced references, like the one on the left of the figure, since it is much easier to compare to that than to an actual picture. Our first Contributions II and I originally were developed with the purpose to test the limits of physically based rendering, testing how close a rendering using path tracing can get to real images. Both investigations led to a number of interesting insights on physically based rendering.

In our first Contribution I, we created a combined appearance model to predict the appearance of cloudy apple juice. The goal here is to be able to predict the final appearance of apple juice by changing production parameters, such as the concentration of the juice or the type of environment the apples are pressed in. The business case would be for a juice company to predict appearance changes due to a change in the concentration of apples in the juice, with the constraint

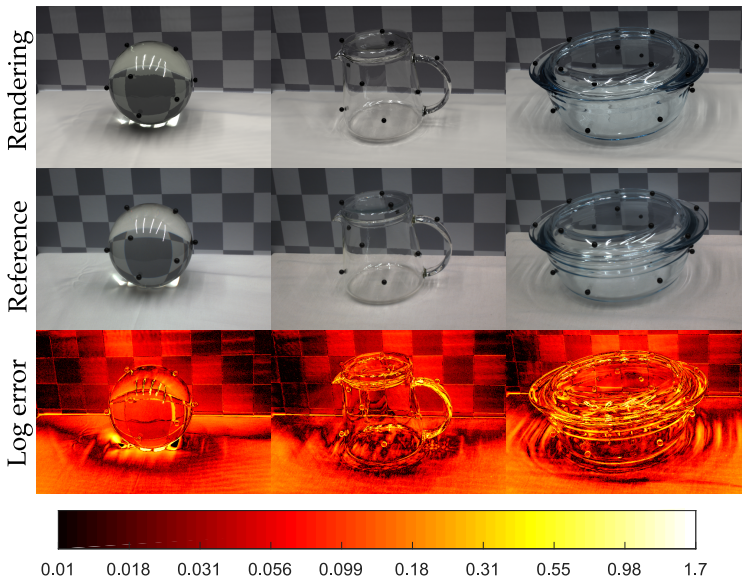


Figure 4.3: Comparing quantitatively renderings (top row) with real images (mid row). Log-error is shown on the bottom row.

that the final product should still appease the customer. Given that the space of production parameters is potentially huge, it is important to give immediate feedback using accurate physically based rendering, so that various possible parameter configurations can be tested quickly. This comparison framework could be used in both directions: as prediction for the final appearance, but also to measure the production parameters of an unknown sample.

A first insight is that if the scene is carefully set and calibrated, a comparison is definitely possible, and actual production parameters can be estimated. We can see this in Figure 4.2, where we compare a patch of the final rendering result with reference. We observed the biggest issue to be in carefully setting and calibrating the scene. In this first proof of concept, we placed the objects and the light in the rendered scene manually, using reasonable estimates for their geometry and appearance parameters. This contribution led to discover a number of different challenges in comparing pictures and renderings, mostly related to the scene. Subtle changes in the scene lead to big differences in direct comparisons, especially when the material influences its surroundings, such as in the case of cloudy apple juice. See the light caustic next to the glass in Figure 4.2. In this initial proof of concept, we simply compared a patch to get a reasonable parameter estimate. In the next Contribution II, we set into improving the whole process of comparison and data reconstruction. From our

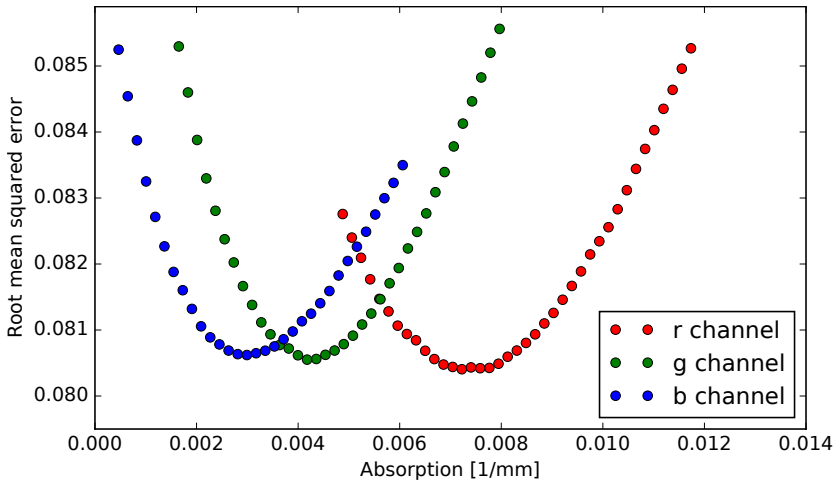
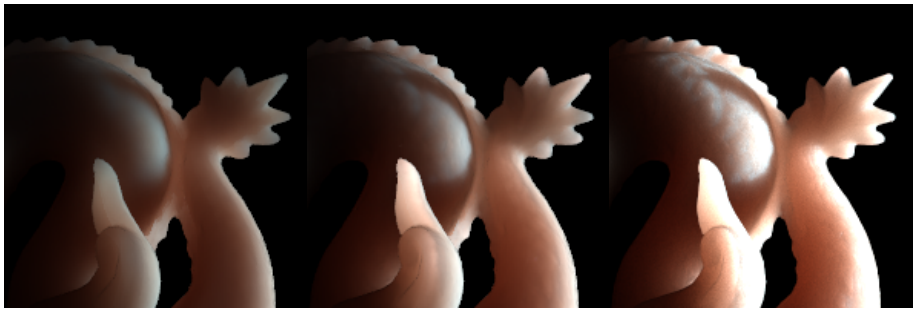


Figure 4.4: Estimating the absorption parameter σ_a for the glass bowl in Figure 4.3. Each coefficient was estimated independently. Each dot in the graph corresponds to a rendered image.

proof of concept in Contribution I, we get a double message. First, it tells us that it is important to validate renderings with photographs, to ensure that the rendering matches the appearance in the real world. Secondly, it tells us that to test or estimate multiple parameter configurations, depending on the application, we need fast rendering. We discuss the first aspect in the rest of this section, referring to the next section for a more complete discussion on fast rendering of scattering materials.

In Contribution II, we strive to improve upon the previous results of comparing rendering with images, for a slightly different application. In this case, we are comparing images of glass objects. As it is true for scattering materials, the appearance of glass objects (see Section 2.3.1) is greatly influenced by the surrounding scene, making the scene estimation arguably even more important for this application. We use a full pipeline to accurately estimate the scene, scan glass objects with CT scanners and place them in the scene for final rendering. Our main contribution of this work is actually the pipeline, that allows researchers to compare pictures and renderings of glass objects, getting a quantitative comparison at the end. Some results are in Figure 4.3. This ability of quantitatively compare images and rendering enables us to further improve existing techniques in acquisition, rendering and reconstruction. As another goal of our improved reconstruction results, we can estimate material properties much more accurately than our results in Contribution I and in Figure 4.2. We measure both the relative index of refraction η and the spectral absorption coefficient σ_a



our method [Jensen et al., 2001] | our method [Frisvad et al., 2014] | unbiased [Frisvad et al., 2014]

Figure 4.5: Comparing our interactive rendering method, with different dipoles and against the unbiased rendering technique reported in Appendix VII.

for each of the glass objects, as we can see in Figure 4.4, where we show the results for the spectral components of the estimated σ_a. Note that each point of the graph is a rendering, further proving the point to use fast rendering to generate a great number of images to estimate the material properties. Since we want accuracy, we need to use unbiased path tracing, that we accelerate using the GPU and execute at lower resolutions to achieve fast and accurate rendering of these images. In this particular case, our renderings are accurate enough for parameter estimation after a few minutes.

4.2 Interactive rendering of scattering media

After discussing parameter estimation in the previous section, in this section we start discussing our first interactive technique. In this section, we acknowledge the fact that for many applications, including previews, games, etc. we can accept lower accuracy in our renderings (see Figure 1.5). In this technique, we achieve a photorealistic look using the BSSRDF approximation presented in Section 2.2.2.

Our Contribution III is a rasterization-based caching scheme to improve photorealism of existing techniques. The most important contribution in this technique is that allows rendering using directional BSSRDFs like the directional dipole discussed in Section 2.2.6. In particular, our technique allows to render with any BSSRDF analytical model that depends on $\vec{\omega}_i$, the direction of the incoming light. In Figure 4.5 we can see a comparison between the standard dipole [Jensen

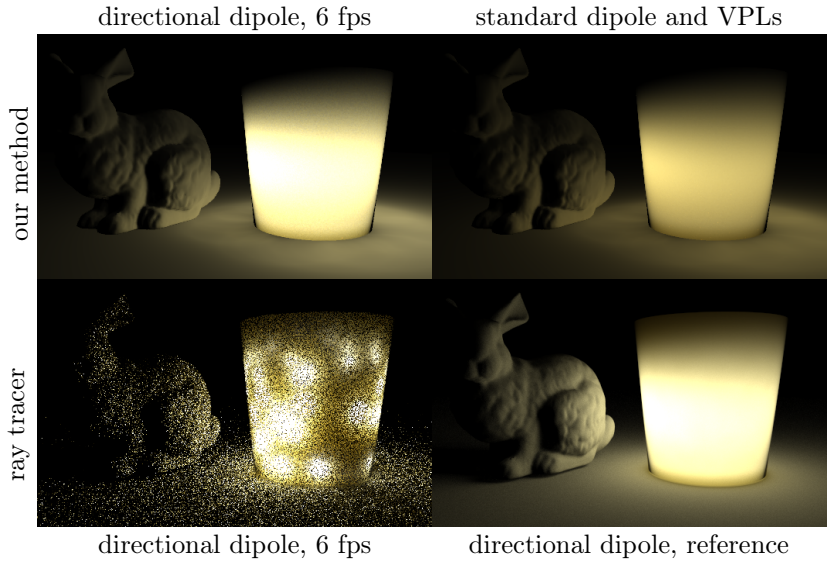


Figure 4.6: Equal time comparison (left column) of our method with the reference method and qualitative comparison with diffuse subsurface scattering (upper right) and the converged reference solution (Appendix VII, lower right). The scene is lit by a point light in a white grapefruit candle holder.

et al., 2001] and the directional dipole [Frisvad et al., 2014] in the first two images, rendered using our technique. With the directional dipole, we can compute more subtle scattering effects, accounting partially for single scattering, that in previous techniques needed to be added separately. Most of the interactive and real-time techniques for rendering BSSRDFs assume that the BSSRDF is function of the distance r between the point of incidence and emergence. This can be exploited for different optimizations, like filtering, precomputation or tabulation, that are not feasible anymore when it comes to using a directional BSSRDF. Our technique is unique in handling this specific type of directional BSSRDFs in the interactive domain.

In the classification of different techniques presented in Chapter 3, our technique is mostly a caching technique, with some filtering required to assemble the final image. Our technique leverages the strengths of rasterization, storing progressive maps of scattered radiosity rendered from different directions around the object. We can now account for the directionality of the light in the computation, and in addition progressively store the intermediate result as soon as the light and the object do not change. The technique is not real-time but interactive (80 to 160 milliseconds on a 2014 GPU), and does not require

neither precomputation nor texture parameterization. Given this features, we can apply this technique to procedural deformable objects, something that is generally difficult to achieve with precomputation techniques. We also proposed an improved sampling scheme, that via a light G-buffer samples always close to the light source, allowing light to propagate through objects and around sharp corners. This can be seen in particular in Figure 4.6, in the top left image, where a point light source is placed inside a candle holder made of a scattering material. Figure 4.5 shows the price we pay for our improved technique, since it introduces some artifacts in the rendering of the directional dipole, visible as dimming. We can see this comparing and showing the difference against the ray tracing based unbiased rendering technique for scattering materials reported in Appendix VII.

As another contribution of our technique, we leverage our scattered radiosity maps to place VPLs on the surface of the objects. This allows to transfer emergent light onto other surfaces. An example can be seen in Figure 4.6, where we transport the light from a point light inside an object on the outside, illuminating the bunny on the tabletop. In the same figure, we can see on how we successfully compare to the technique in Appendix VII, that in the same time cannot achieve the same results, looking non converged and noisy (bottom left).

To sum up, the take home message of this technique is that improved physical models, such as the directional dipole, sometimes do not allow us to reuse previous work. So, we need to develop new techniques to achieve interactive result, so that we can include these more accurate effects. Moreover, the benefits of caching data in this case become particularly apparent, since they allow us to recycle the stored radiosity for another technique to use.

4.3 Interactive stable ray tracing

After dealing with subsurface scattering, we continue exploring the spectrum of interactive photorealistic techniques with stable ray tracing (Contribution IV). As in the previous section, we use caching as a technique to improve existing physically based techniques. In the previous section, we created a technique that anchors the scattering contribution to the surface of the object using depth-augmented scattered radiosity maps. In stable ray tracing, we solve a completely different problem, but our technique in the essence has the same overall purpose, caching points on surfaces. In this contribution, in particular, we track these points in order to achieve temporally stable renderings.

Supersampling, 2 spp
sharpness: 0.7924

Supersampling, 2 spp
+ temporal anti-aliasing
sharpness: 0.5348

Supersampling, 32 spp
sharpness: 0.6771

Stable RT, 1 spp
sharpness: 0.7085

Stable RT, 1 spp
+ temporal integration
sharpness: 0.6060

Supersampling, 32 spp
sharpness: 0.6771

Figure 4.7: Different techniques on a frame in the Sponza video. Equal time comparison apart from the reference in last column (32 spp). Sharpness is also shown (higher is sharper). Stable RT shows lower noise (column 1) compared to reference (column 3). With temporal techniques on top (column 2) stable RT is sharper.

Our technique was developed with interactive ray tracing in mind. In interactive or real-time ray tracing, the number of rays we can trace per pixel becomes extremely limited, in the order of one or two full paths. Also, the shading locations change every frame, causing a noisy image, both spatially and temporally. Existing techniques, namely temporal anti aliasing [Karis, 2014], can mitigate the problem, though they generally introduce blurring. In our technique, we recycle shading locations across frames using a screen space data structure. By shading always the same points, we improve temporal stability, while retaining sharpness as well. We achieve this at interactive frame rates (20-60 milliseconds per frame). We can see the results in Figure 4.7. In the result we can see how stable ray tracing yields a sharper result than temporal anti-aliasing. The technique contributes as a rendering system to improve temporal stability: other existing

techniques can be further applied in top to improve shading quality.

Another contribution of this technique is that it shows how we can leverage the strength of one interactive technique, namely ray tracing, against the most commonly used rasterization. Current hardware does not allow shading locations to be arbitrarily chosen within a pixel, relying on fixed patterns. In our technique, we allow shading locations to vary in screen space per pixel, while staying the same in world space.

Finally, another advantage of this technique, in particular in a photorealistic rendering context, is that it enables us to store information across frames. In the paper, we present an example case where we store indirect path traced illumination. This will allow in the future to extend the technique with other sort of data, to further improve the technique. This gives the take home message of this technique: we need generic, robust techniques that can be applied in a variety of situations, inexpensive and that can work together with existing techniques. We believe that stable ray tracing satisfies these characteristics.

A particular application where the stable ray tracing caching scheme would be particularly useful is virtual reality. Because of the close vicinity of the eye to the display and the perception system in our eyes, temporal stability issues are particularly obvious. Unfortunately, given the real-time requirements for virtual reality this technique is not applicable at the moment.

4.4 Applying interactive photorealistic techniques

In the final paper we discuss in Contribution V, we discuss a proof of concept for virtual reality rendering of physically based materials. In this context, applications need to consistently perform at 90 frames per second or faster, to avoid issues for the users, e.g. dizziness or motion sickness. In our application, we modify the rendering pipeline of the Unity game engine to include measured BRDFs, in the discretized form of the MERL database, as described in Section 2.3.2.2. The application provides a painting environment for artists to paint on object surfaces using measured BRDFs, using a HTC Vive headset and controllers. The environment includes the possibility to move and rotate objects, paint with different sized brushes, a movable light and an undo button. The direct illumination is given by a point light, plus low frequency environment illumination using spherical harmonics. A reflection map is added on top, with intensity depending on the specularity of the material, and depending on the ratio between peak and average value of the BRDF. The overall simulation renders at steady 11 milliseconds per frame.

Figure 4.8: Pictures illustrating our VR demo application, with an in-game screenshot (left) and a picture of the setup (right).

This proof of concept was born as an inspection tool to debug physically based materials, aided by the provided in-game controllers (see an example image in Figure 4.8). The application shows once more how important it is to achieve interactive photorealistic rendering in virtual reality, giving a glimpse on how photorealistic materials are able to augment the immersiveness of the application.

4.5 Discussion

In the previous sections, we showed how our contribution overall contribute to the goal of expanding interactive technique towards a more physically accurate framework. The natural step of the topics discussed in this thesis is to further expand the techniques in photorealism, by including as an example more accurate physical models, but retaining the same performance level. In the following, we discuss some possible avenues of expansion of our work.

Validating path tracing. Some avenues are possible in continuing the work we initiated in Contributions I and II. In particular, we would like to continue our initial goal of validating path tracing. This would require expanding the technique to be more accurate, in particular in regards to geometry. For a more complete validation, we would like to expand the technique to handle fully scattering materials instead of glass only. Other avenues of research are possible in regards of estimating parameters: scattering parameters (σ_a, σ_s and g) are obvious candidates for a further expansion.

Hybrid rasterization-ray tracing rendering techniques. As for now, we focused on our technique to be exclusively rasterization or ray tracing based. Some potential improvements can be thought by combining the two techniques. For example, some could think of an extension of our stable ray tracing in Contribution IV to support our technique from Contribution III. In this particular case, the scattered radiosity maps can be replaced by the stable ray traced points. The rasterization of the light G-buffer would still have to be executed, making this a hybrid technique. Another example, in our virtual reality contribution V, would be to evaluate reflections via ray tracing, to properly include the overall contribution instead of an approximation via a precomputed probe.

Virtual reality. In recent year, virtual reality has become more prominent in real-time applications. Given the hard constraints of virtual reality, the techniques we developed would need to be adapted to fit a virtual reality environment. In particular our stable ray tracing algorithm in Contribution IV seems like a good fit for a virtual reality environment, that is particularly plagued by temporal stability issues.

Dataset generation for machine learning. Interactive techniques for photorealistic rendering allow generating image data much faster than traditional techniques. This makes them particularly useful to generate synthetic data to train image based machine learning techniques.

Chapter 5

Conclusion

In this thesis, we have presented the results of the past three year of Ph.D. studies, that started with the goal of developing new techniques for interactive rendering, with an eye onto interactive photorealism. The exploration of different techniques during the Ph.D. studies did not follow a predefined scheme. The individual contributions have been developed as consequence of the findings along the way. We are happy that we managed to keep our contributions along a common theme, bringing photorealistic rendering into the interactive domain.

Over the course of the Ph.D. studies we contributed with a number of results and publications. These results are relevant in many fields, including product visualization, architectural rendering, interactive previews, previews of rendering results, and video game production. We have presented a range of publications that contribute with techniques that can be employed in these fields, addressing various important challenges in each area.

In recent years, the real-time rendering community is pushing more and more towards physically based models. Given this, we can use clever techniques to introduce additional photorealism in existing techniques. We saw a need for more accurate predictive rendering in an industrial domain in Contribution I, where we tied industrial parameters to the rendering of physically based apple juice. We first investigated the need for photorealism to validate existing reconstruction and acquisition techniques in Contribution II. Moreover, we proved

the need of accurate photorealistic rendering to measure unknown radiometric parameters.

After this excursion in high accuracy photorealistic rendering, we moved into developing techniques that improve upon existing algorithms to further enhance physically based models. In our Contribution III, we contributed with the first technique able to render directional BSSRDF models on deformable objects, allowing also to transport emergent scattered light across the scene. Inspired by our industrial lookout, we set on solving the problem of temporally stable global illumination, in the growing field of interactive ray tracing. We contributed with an interactive technique in Contribution IV that allows temporally stable sharp global illumination, and that can be easily added on top of many existing algorithms. Finally, in Contribution V we contributed with a proof of concept to the growing field of virtual reality rendering, introducing physically based measured BRDFs in a real-time environment with hard constraints.

All the contributions listed above contributed to achieve new insights in the individual areas of interest. We can summarize the highlights of the single contributions in this thesis as the following:

- Investigated the challenges in comparing images with photorealistic renderings (Contributions I, II).
- Proposed a reconstruction and assembly pipeline that allows to compare images to renderings of the same scene (Contribution II).
- Created a new dataset of transparent objects scene and CT scan data (Contribution II).
- Proposed new techniques to estimate material parameters (Contributions I, II).
- Explored the challenges of fast interactive photorealistic rendering for rendering and quality assurance (Contributions I,III).
- Demonstrated the first interactive application of light-directional subsurface scattering (Contribution III).
- Developed a new interactive ray tracing technique to improve temporal stability without sacrificing sharpness (Contribution IV).
- Created new interactive rendering techniques to improve photorealistic light transport (Contributions III, IV).
- Created a novel Virtual Reality application to showcase physically based materials in a hard real-time context (Contribution V).

Based on these contributions, we conclude that the goal of developing new interactive techniques that push the boundaries of photorealistic rendering in the interactive and real-time domain has been achieved.

Appendix I

Interactive Appearance Prediction for Cloudy Beverages

Workshop on Material Appearance Modeling (2016)
H. Rushmeier and R. Klein (Editors)

Interactive Appearance Prediction for Cloudy Beverages

Alessandro Dal Corso[1], Jeppe Revall Frisvad[1], Thomas Kim Kjeldsen[2], and Jakob Andreas Bærentzen[1]

[1]Technical University of Denmark
[2]Alexandra Institute, Denmark

Abstract

Juice appearance is important to consumers, so digital juice with a slider that varies a production parameter or changes juice content is useful. It is however challenging to render juice with scattering particles quickly and accurately. As a case study, we create an appearance model that provides the optical properties needed for rendering of unfiltered apple juice. This is a scattering medium that requires volume path tracing as the scattering is too much for single scattering techniques and too little for subsurface scattering techniques. We investigate techniques to provide a progressive interactive appearance prediction tool for this type of medium. Our renderings are validated by qualitative and quantitative comparison with photographs. Visual comparisons using our interactive tool enable us to estimate the apple particle concentration of a photographed apple juice.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—

1. Introduction

Quoting a food science article: "the visual appearance of a cloudy drink is a decisive factor for consumer acceptance" [Bev02]. We therefore believe that appearance prediction is highly relevant in the beverage industry of the future. It is important to predict the visual effect of a production parameter being modified, so that production parameters may be optimized without negative impact on the visual quality of the product. An appearance model also potentially enables analysis of product properties using camera sensors.

rendering photograph

Figure 1: *Cloudy apple juice photographed and rendered using our appearance model. In the model, apple particle concentration (0.8 g/l) and apple storage period (4 days) were selected to match the photograph.*

The cloudy part of a beverage typically consists of oil droplets or fruit flesh. We use Lorenz-Mie theory to go from a particle size distribution of fruit flesh particles, for example, to the scattering properties that we would use in a volume rendering [FCJ07]. Once the scattering properties are available, there are many ways to render the medium. However, in the case of a cloudy beverage, the scattering is neither low enough for single scattering techniques nor high enough for subsurface scattering techniques. We thus use full volume path tracing [Rus88], which quite accurately predicts the appearance of milk [FCJ12].

The complex refractive indices ($n = n' + in''$) of host liquid and particle inclusions are required as input for the Lorenz-Mie theory. In many cases, these parameters depend to some degree on production parameters. In addition, the concentration of particles is often a production parameter. This enables us to build appearance models that are parameterized by production parameters. As a case study, we build such an appearance model for apple juice.

The ability to quickly test the visual influence of different parameters is one of the key advantages of material appearance prediction. This advantage is however hampered by the very long rendering times that we typically experience with full path tracing of multiple scattering in a volume. To have fast visual feedback and shorter rendering times, we use progressive path tracing implemented on the GPU using the OptiX framework [PBD*10]. We shortly cover the steps that we took to obtain an interactive appearance prediction tool based on this framework.

We validate our appearance model by qualitative comparison of rendered results with a photograph of a commercial unfiltered apple juice (Figure 1). As the visual similarity of rendered and photographed juice is better with some parameter settings than others, we can use our appearance model to roughly estimate what the parameters might have been during production of the photographed juice.

2. Appearance Model

The model we present for apple juice is intended for unfiltered press juices. Consequently, we model it simply as apple particles in clear apple juice. To obtain complex refractive indices for these constituents, we exploit that the imaginary part is directly related to the absorption of the material [FCJ07]. The absorption of the apple particles and juice both depend on how the apples are handled, and on whether anything is done to prevent the enzymatic browning that naturally takes place when apples are peeled, bruised, or pressed in an oxidative environment.

Host. Fruit juices have a significant amount of dissolved solids (mostly sugars [HRC09]). The total concentration of soluble solids X influences the real part of the refractive index of the host. This concentration is typically measured in degrees Brix (°Bx), which is the weight percentage of dissolved solids, and it depends on the ripeness of the harvested fruits. More ripe fruits have a higher sugar content. We use two measurements from Genovese and Lozano [GL06] to find a correction for the real part of the refractive index of water. After correction, the real part of the host refractive index is

$$n'_{\text{host}}(\lambda) = n'_{\text{water}}(\lambda) + 0.0016 \frac{\text{dl}}{\text{g}} X\,,$$

where λ is the wavelength of the light (*in vacuo*). For the imaginary part of the refractive index, we use the different absorbance spectra of browned clarified apple juice reported by Beveridge et al. [BFH86, Fig. 4A]. As the wavelength increases, the absorbance values become smaller and more uncertain. Since they should not decrease below the absorption of water, we convert them to absorption coefficients and gradually blend them with the absorption spectrum of water in the range of wavelengths from 400 nm to 700 nm. The spectra we get for n''_{host} as a result are in Table 1.

λ [nm]	n''_{host} 4 days	n''_{host} 9.5 days peeled	n''_{host} 9.5 days	n''_{host} 27 days	n''_{part}
375	$1.58 \cdot 10^{-6}$	$1.03 \cdot 10^{-6}$	$2.20 \cdot 10^{-6}$	$3.20 \cdot 10^{-6}$	$1.14 \cdot 10^{-5}$
400	$1.69 \cdot 10^{-6}$	$1.10 \cdot 10^{-6}$	$2.35 \cdot 10^{-6}$	$3.41 \cdot 10^{-6}$	$1.02 \cdot 10^{-5}$
425	$6.78 \cdot 10^{-7}$	$8.92 \cdot 10^{-7}$	$1.39 \cdot 10^{-6}$	$2.78 \cdot 10^{-6}$	$1.08 \cdot 10^{-5}$
450	$4.47 \cdot 10^{-7}$	$8.25 \cdot 10^{-7}$	$1.17 \cdot 10^{-6}$	$2.68 \cdot 10^{-6}$	$1.15 \cdot 10^{-5}$
475	$4.08 \cdot 10^{-7}$	$7.18 \cdot 10^{-7}$	$9.79 \cdot 10^{-7}$	$2.19 \cdot 10^{-6}$	$9.83 \cdot 10^{-6}$
500	$3.97 \cdot 10^{-7}$	$5.50 \cdot 10^{-7}$	$7.33 \cdot 10^{-7}$	$1.44 \cdot 10^{-6}$	$8.36 \cdot 10^{-6}$
525	$3.51 \cdot 10^{-7}$	$3.65 \cdot 10^{-7}$	$5.06 \cdot 10^{-7}$	$8.71 \cdot 10^{-7}$	$3.59 \cdot 10^{-6}$
550	$2.53 \cdot 10^{-7}$	$2.43 \cdot 10^{-7}$	$3.04 \cdot 10^{-7}$	$5.05 \cdot 10^{-7}$	$2.54 \cdot 10^{-6}$
575	$1.67 \cdot 10^{-7}$	$1.67 \cdot 10^{-7}$	$1.84 \cdot 10^{-7}$	$3.09 \cdot 10^{-7}$	$1.92 \cdot 10^{-6}$
600	$1.06 \cdot 10^{-7}$	$1.13 \cdot 10^{-7}$	$1.13 \cdot 10^{-7}$	$1.90 \cdot 10^{-7}$	$1.52 \cdot 10^{-6}$
625	$6.78 \cdot 10^{-8}$	$8.22 \cdot 10^{-8}$	$7.64 \cdot 10^{-8}$	$1.18 \cdot 10^{-7}$	$1.41 \cdot 10^{-6}$
650	$4.74 \cdot 10^{-8}$	$6.23 \cdot 10^{-8}$	$5.24 \cdot 10^{-8}$	$7.22 \cdot 10^{-8}$	$1.47 \cdot 10^{-6}$
675	$3.70 \cdot 10^{-8}$	$4.63 \cdot 10^{-8}$	$3.91 \cdot 10^{-8}$	$4.58 \cdot 10^{-8}$	$1.78 \cdot 10^{-6}$
700	$3.48 \cdot 10^{-8}$	$3.48 \cdot 10^{-8}$	$3.48 \cdot 10^{-8}$	$3.48 \cdot 10^{-8}$	$6.13 \cdot 10^{-7}$
725	$8.59 \cdot 10^{-8}$	$8.59 \cdot 10^{-8}$	$8.59 \cdot 10^{-8}$	$8.59 \cdot 10^{-8}$	$2.08 \cdot 10^{-7}$
750	$1.47 \cdot 10^{-7}$	$1.47 \cdot 10^{-7}$	$1.47 \cdot 10^{-7}$	$1.47 \cdot 10^{-7}$	$1.78 \cdot 10^{-7}$
775	$1.49 \cdot 10^{-7}$	$1.49 \cdot 10^{-7}$	$1.49 \cdot 10^{-7}$	$1.49 \cdot 10^{-7}$	$1.58 \cdot 10^{-7}$

Table 1: *Absorption of clarified apple juice (host) [BFH86] and apple flesh (particles) [LCHA10] reported in the form of the imaginary part of the index of refraction.*

Particles. We model the apple particles as browned apple flesh. The real part of the refractive index of the apple particles was estimated by Benitez et al. [BGL07] to be $n'_{\text{part}}(\lambda) = 1.487$. For the imaginary part, we use the absorption spectrum measured by Lu et al. [LCHA10] for bruised apple tissue (two days after bruising). These measurements are in the range of wavelengths from 500 nm and up. For the shorter wavelengths, we use the absorption spectrum measured for apple flesh by Saeys et al. [SVRT*08]. The two curves fit fairly well together, so we assume that the bruising mostly affects absorption at wavelengths from 500 nm and up. The combined spectrum that we use is in Table 1. The size distribution of the apple particles is typically bimodal [Bev02]. Smaller particles (diameter of less than around 0.65 μm) will stay suspended, while larger particles will sediment during storage and will require the juice to be shaken to be resuspended. We use a bimodal particle size distribution measured for a centrifuged apple juice [ZPDG94, Bev02].

3. Rendering Method

Our outset is unidirectional path tracing [Kaj86, PH10]. We use Russian roulette to terminate paths probabilistically (based on material absorption) and to choose between reflection and refraction in the case of transparent materials [AK90]. Our scene is composed of four different kinds of materials: emissive, diffuse, glass, and liquid. For accurate rendering, we assign liquid/air, glass/air, and liquid/glass interfaces to the specular surfaces in our scene.

When path tracing specular surfaces, we use the laws of reflection and refraction with Fresnel reflectance as the probability of reflection. To account for glass absorption, we use an RGB absorption coefficient for crown glass $\sigma_{a,\text{glass}} = (1.87, 1.86, 3.01)\,\text{m}^{-1}$ that we calculated from n'' of Schott N-K5 glass. Since the drinking glass is not transmittance-

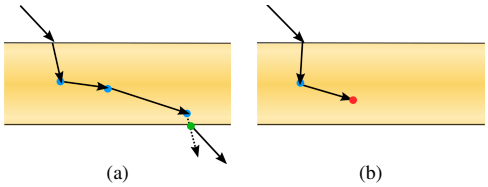


Figure 2: *The scattering process using shadow rays. Each shadow ray has a sampled direction (sampled with the Henyey-Greenstein phase function) and length (from an exponential distribution of average σ_t). The random walk either (a) exits the material and the path continues or (b) suffers an absorption event and the path is terminated.*

optimized optical glass, we scaled the absorption coefficient by 8. The average over the color bands of the beam transmittance $T_r = e^{-\sigma_{a,\text{glass}}\, s}$, where s is the distance traveled through the glass medium, is then the probability that the path continues through the glass without being terminated.

The cloudy beverage is considered a specular material, but it also contains light scattering particles. When a ray passes through this medium (after refracting into it or after internal reflection), we perform a volumetric scattering process as illustrated in Figure 2. This is done by a stochastic walk inside the medium based on the scattering properties (scattering and extinction coefficients, σ_s and σ_t, and asymmetry parameter, g) that we obtain from the Lorenz-Mie theory. At each step of the walk, we use the scattering albedo σ_s/σ_t in a Russian roulette as the probability of the path surviving without being absorbed. We discard the whole walk if the ray is absorbed. If the ray is not absorbed, we sample the distance to the next scattering event using $s = -\ln(\xi)/\sigma_t$, where $\xi \in (0,1)$ is a uniform random variable. If s is beyond the surface of the medium, the next scattering event is interaction with the surface in the usual way. If not, we sample a new scattering direction using the Henyey-Greenstein phase function [PH10] (which is a function of g). To improve efficiency, we randomly pick one RGB components when performing the scattering process, and all tracings inside the medium are done with shadow rays using the distance to the next scattering event as the maximum trace distance.

4. Materials

To capture a reference photograph, we set up a scene consisting of a drinking glass with unfiltered apple juice placed on a neutral white surface with a neutral white background. The glass was illuminated by a large diffuse light source at a 45° angle. We used a standard digital single-lens reflex (DSLR) camera with a 50 mm lens. The light source was a Bowens BW3370 100W Unilite, which is a compact fluorescent light source with a correlated color temperature (CCT) of 6400 K. This is fairly close to the equal energy point

	4 days	9.5 days peeled	9.5 days	27 days
0.0 g/l	0.1229	0.1190	0.1271	0.1568
0.1 g/l	0.0583	0.0622	0.0836	0.1311
0.2 g/l	0.0457	0.0570	0.0821	0.1316
0.5 g/l	0.0352	0.0561	0.0832	0.1338
1.0 g/l	0.0332	0.0537	0.0799	0.1307
2.0 g/l	0.0376	0.0494	0.0711	0.1211

Table 2: *RMSE for a patch of color in the lower part of the glass. We note a minimum error around the 0.5-1.0 g/l concentration and 4 days storage time.*

(E with $x = y = 1/3$) in the chromaticity diagram, which is also the reference white point of the CIE RGB color space. We thus model our light source as being purely white, and convert the spectral optical properties obtained from our appearance model to CIE RGB using the RGB color matching functions reported by Stockman and Sharpe [SS00].

The apple juice is an unfiltered press juice from Orskov Foods with a sugar content of 10 g per 100 ml. We estimated $X = 11.25$ g/dl as the juice also contains pectin, organic acids, and salts in addition to the sugar [GL06], and with this X we get n'_{host} equal to one previously measured for regular pressed cloudy apple juice [BGL07].

5. Results

We rendered our scene with various model parameter settings. These renderings are in Figure 3 and were progressively updated for 5,000 frames. We used this data material for comparing with the photograph in Figure 1. In a qualitative visual comparison, we found that the photograph most closely resembles renderings with concentration in the 0.5-1.0 g/l range and a storage time of 4 days. For quantitative comparison, we used an image patch covering the lower part of the glass and the nearby part of the caustic. As a metric, we used root-mean-squared error (RMSE). The results are in Table 2. These measurements place the concentration between 0.5 g/l and 1.0 g/l, slightly leaning towards the latter and confirming our qualitative comparison. In the renderings, visual effects due to total internal reflection are too prominent toward the sides in the upper part of the glass. This part of the modeled glass is probably too thin.

Our implementation of the method, based on OptiX [PBD*10], runs progressively on the GPU, with an average frame rate of 12 frames per second on an NVIDIA GeForce 780 Ti card. This enables us to get nearly converged results in less than 10 minutes. A visual comparison of the main liquid is possible within a minute (~600 frames), while full convergence takes much longer (10^5 frames in Figure 1).

6. Discussion

By combining existing techniques in our framework, we enable interactive testing of the influence of different production

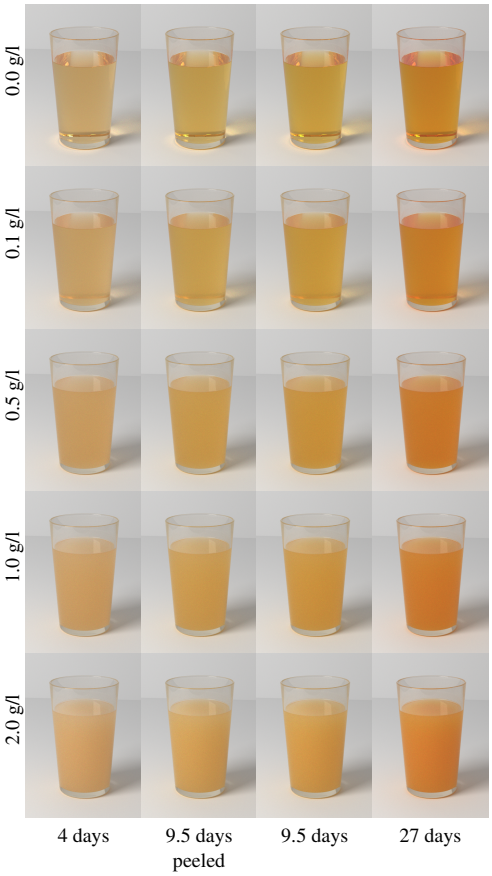


Figure 3: *Renderings with two parameters being varied: particle concentration and apple storage period.*

parameters on the appearance of a cloudy beverage. Our case study enables approximate simulation of the appearance of cloudy apple juice during production. However, we cannot match the photograph perfectly, and can thus only give a rough estimate of the parameters. As a possible future step, we want to improve precision by better estimating scene geometry, camera parameters, and lighting environment.

Acknowledgments. We are grateful to Anders Wang Kristensen and Marek Krzysztof Misztal who helped model the 3D scene and capture the reference image.

References

[AK90] ARVO J., KIRK D.: Particle transport and image synthesis. *Computer Graphics (Proceedings of SIGGRAPH '90) 24*, 4 (August 1990), 63–66. 2

[Bev02] BEVERIDGE T.: Opalescent and cloudy fruit juices: Formation and particle stability. *Critical Reviews in Food Science and Nutrition 42*, 4 (2002), 317–337. 1, 2

[BFH86] BEVERIDGE T., FRANZ K., HARRISON J. E.: Clarified natural apple juice: Production and storage stability of juice and concentrate. *Journal of Food Science 51*, 2 (March 1986), 411–433. 2

[BGL07] BENITEZ E. I., GENOVESE D. B., LOZANO J. E.: Scattering efficiency of a cloudy apple juice: Effect of particles characteristics and serum composition. *Food Research International 40*, 7 (August 2007), 915–922. 2, 3

[FCJ07] FRISVAD J. R., CHRISTENSEN N. J., JENSEN H. W.: Computing the scattering properties of participating media using Lorenz-Mie theory. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (July 2007), 60:1–60:10. 1, 2

[FCJ12] FRISVAD J. R., CHRISTENSEN N. J., JENSEN H. W.: Predicting the appearance of materials using Lorenz-Mie theory. In *The Mie Theory: Basics and Applications*, Hergert W., Wriedt T., (Eds.), vol. 169 of *Springer Series in Optical Sciences*. July 2012, ch. 4, pp. 101–133. 1

[GL06] GENOVESE D. B., LOZANO J. E.: Contribution of colloidal forces to the viscosity and stability of cloudy apple juice. *Food Hydrocolloids 20*, 6 (August 2006), 767–773. 2, 3

[HRC09] HUANG Y., RASCO B. A., CAVINATO A. G.: Fruit juices. In *Infrared Spectroscopy for Food Quality Analysis and Control*, Sun D.-W., (Ed.). Academic Press/Elsevier, 2009, ch. 13, pp. 355–375. 2

[Kaj86] KAJIYA J. T.: The rendering equation. *Computer Graphics (Proceedings of SIGGRAPH 86) 20*, 4 (August 1986), 143–150. 2

[LCHA10] LU R., CEN H., HUANG M., ARIANA D. P.: Spectral absorption and scattering properties of normal and bruised apple tissue. *Transactions of the American Society of Agricultural and Biological Engineers 51*, 1 (2010), 263–269. 2

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: OptiX: a general purpose ray tracing engine. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010) 29*, 4 (July 2010), 66:1–66:13. 2, 3

[PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*, second ed. Morgan Kaufmann/Elsevier, 2010. 2, 3

[Rus88] RUSHMEIER H. E.: *Realistic image synthesis for scenes with radiatively participating media*. PhD thesis, Cornell University, Ithaca, NY, USA, 1988. 1

[SS00] STOCKMAN A., SHARPE L. T.: The spectral sensitivities of the middle- and long-wavelength-sensitive cones derived from measurements in observers of known genotype. *Vision Research 40*, 13 (2000), 1711–1737. 3

[SVRT*08] SAEYS W., VELAZCO-ROA M. A., THENNADIL S. N., RAMON H., NICOLAI B. M.: Optical properties of apple skin and flesh in the wavelength range from 350 to 2200 nm. *Applied Optics 47*, 7 (March 2008), 908–919. 2

[ZPDG94] ZIMMER E., PECORONI S., DIETRICH H., GIERSCHNER K.: Process technological and chemical basis of the production of cloudy apple juice, also considering continuous processes. I. Contributions to the chemical analysis of natural cloudy apple juices including physical parameters. Part II. *Die industrielle Obst- und Gemüseverwertung 79*, 12 (1994), 426–434. 2

Appendix II

Scene reassembly after multimodal digitization and pipeline evaluation using photorealistic rendering

Scene reassembly after multimodal digitization and pipeline evaluation using photorealistic rendering

JONATHAN DYSSEL STETS[1,†], ALESSANDRO DAL CORSO[1,†], JANNIK BOLL NIELSEN[1], RASMUS AHRENKIEL LYNGBY[1], SEBASTIAN HOPPE NESGAARD JENSEN[1], JAKOB WILM[1], MADS BRIX DOEST[1], CARSTEN GUNDLACH[2], EYTHOR RUNAR EIRIKSSON[1], KNUT CONRADSEN[1], ANDERS BJORHOLM DAHL[1], JAKOB ANDREAS BÆRENTZEN[1], JEPPE REVALL FRISVAD[1,*], AND HENRIK AANÆS[1]

[1] *Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads, 2800 Kongens Lyngby, Denmark*
[2] *Department of Physics, Technical University of Denmark, Fysikvej, 2800 Kongens Lyngby, Denmark*
[†] *Joint primary authors*
[*] *Corresponding author: jerf@dtu.dk*

Compiled May 15, 2018

Transparent objects require acquisition modalities that are very different from the ones used for objects with more diffuse reflectance properties. Digitizing a scene where objects must be acquired with different modalities, requires scene reassembly after reconstruction of the object surfaces. This reassembly of a scene that was picked apart for scanning seems unexplored. We contribute with a multimodal digitization pipeline for scenes that require this step of reassembly. Our pipeline includes measurement of bidirectional reflectance distribution functions (BRDFs) and high dynamic range (HDR) imaging of the lighting environment. This enables pixelwise comparison of photographs of the real scene with renderings of the digital version of the scene. Such quantitative evaluation is useful for verifying acquired material appearance and reconstructed surface geometry, which is an important aspect of digital content creation. It is also useful for identifying and improving issues in the different steps of the pipeline. In this work, we use it to improve reconstruction, apply analysis by synthesis to estimate optical properties, and to develop our method for scene reassembly.

OCIS codes: *(150.4232) Multisensor methods; (150.6910) Three-dimensional sensing; (150.1488) Calibration; (160.4760) Optical properties; (290.1483) BSDF, BRDF, and BTDF; (330.1690) Color.*

https://doi.org/10.1364/AO.56.007679

1. INTRODUCTION

Several research communities work on techniques for optical acquisition of physical objects and their appearance parameters [1–5]. Thus, we are now able to acquire nearly any type of object and perform a computer graphics rendering of nearly any type of scene. The range of applications is broad and includes movie production [2], cultural heritage preservation [3], 3D printing [4], and industrial inspection [5]. A gap left by these multiple endeavors is a coherent scheme for acquiring a scene consisting of several objects that have very different appearance parameters, together with the reassembly of a digital replica of such a scene. Our objective is to fill this gap for the combination of transparent and opaque objects, as many real world scenarios exhibit this combination. An example is a living room, like the one rendered in Fig. 1 (right). We propose a pipeline for acquiring and reassembling digital scenes from this type of heterogeneous real-world scenes. In addition, our pipeline closes the loop by rendering calibrated images of the digital scene that are commensurable with photographs of the original physical scene (see Fig. 1, left). This allows for validation and fine-tuning of appearance parameters. The quantitative evaluation we get from pixelwise comparison of rendered images with photographs is a great improvement with respect to validation of the acquired digital representation of the physical objects.

When addressing the problem of acquiring a heterogeneous scene, there is an infinite variety of scenes and object types to choose from. So, to make our task feasible, we focus on scenes that combine glassware and non-transparent materials, more specifically, white tablecloth and cardboard with a checkerboard pattern. We made these choices as glass requires a different acquisition modality, the tablecloth BRDF is spatially uniform but not necessarily simple, and the cardboard has simple two-color variation. The latter is particularly useful for observing how light refracts through the glass. The chosen case is also of particular interest, since glass is present in many intended

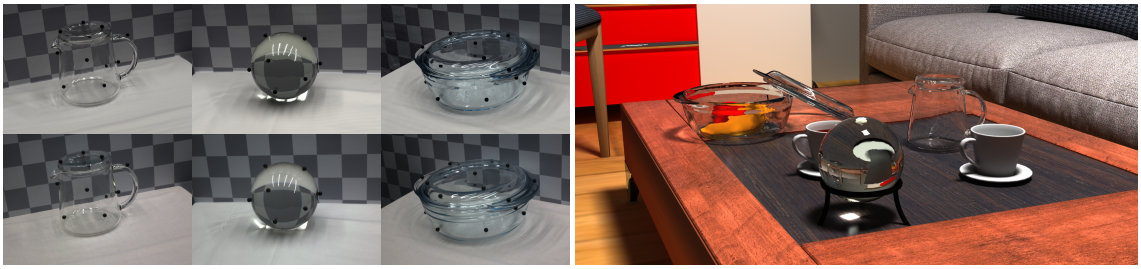


Fig. 1. To the left, we compare rendered images (top) with photographs (bottom). More views are available in supplementary Visualization 1. The scenes to the left were digitized using our pipeline and include both glass objects and non-transparent objects (tablecloth and backdrop). To the right, we exemplify the use of our pipeline for virtual product placement using our digitized glass objects, with estimated optical properties and artifact-reduced removal of markers.

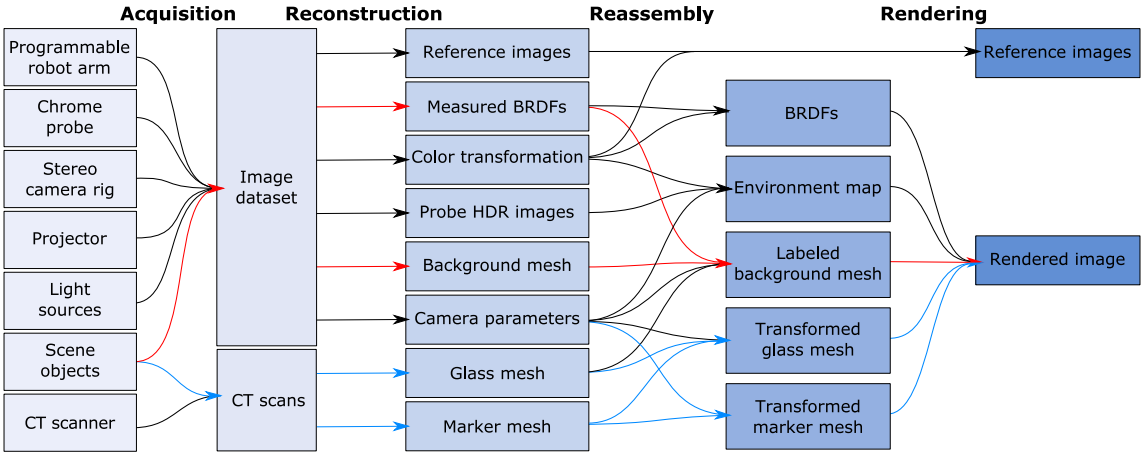


Fig. 2. Overview of our digitization pipeline in four main stages: acquisition, reconstruction, reassembly, and rendering. A video presentation of our pipeline is available in supplementary Visualization 2. Colored arrows show the path of transparent (blue) and non-transparent (red) objects through the pipeline.

applications of optical 3D acquisition. Considering the highly multidisciplinary nature of our work, we have released our dataset.[1] This facilitates further investigation by other researchers of the different steps of our pipeline with the possibility of a quantitative feedback at the end of the process.

A. Related Work and Contributions

Researchers occasionally compare renderings with photographs to provide a qualitative verification of a presented rendering technique. The work by Phong [6], Goral et al. [7], and Takagi et al. [8] are early examples of this trend. A procedure to bring a rendered image close to a photograph was first presented by Meyer et al. [9]. In this work, likeness of images was evaluated perceptually by human observers. Pixelwise comparison of photographs with rendered images is surprisingly uncommon. The few examples we have found are by Rushmeier et al. [10], Karner and Prantl [11], Pattanaik et al. [12], and Jones and Reinhart [13, 14]. These examples build on the rendering framework described by Greenberg et al. [15]. Employing such a framework for more complex scenes is a long and tedious process [16]. The key issue is that a scene specification is expected as an input.

Several problems arise as a result of not having correspondence between the physical and the digital scene. Misalignment due to inaccurate scene and viewing geometry and inaccurate orientation of the lighting environment are some of the essential problems identified in previous work [17, 18]. One way to deal with this problem is to calculate error for image patches when evaluating results [13, 19, 20]. As opposed to this, our digitization pipeline (Fig. 2) provides both reference photographs and correspondingly calibrated scene and viewing geometry so that pixelwise comparison becomes meaningful.

Pixelwise comparison of rendered images with photographs is not only useful for quantifying the photorealism of a rendering in terms of error measurements. We find it particularly useful for improving the digitization pipeline. The fact that our pipeline enables quantitative evaluation led us to more specific contributions in its different steps. These contributions are mostly in the reassembly and are as follows. (a) A cross-modality marker-based placement approach, enabling accurate placement of objects scanned with one modality into scenes scanned with another modality. (b) A soft object deformation technique dea-

[1] Link to appear upon publication.

ling with surface intersections after object placement, which is critical for scenes containing transparent or translucent objects. (c) A micropolygon labeling approach for assigning BRDFs to acquired geometry. (d) A color calibration scheme enabling use of spectral optical properties for calculating reflectance, transmittance, and absorption. (e) Perspective unwrapping of mirror probe images to improve precision when the environment is not very distant. (f) Use of analysis by synthesis for fine-tuning physics-based optical properties.

Digitization is most often unimodal and tailored toward objects with a specific type of surface reflectance behavior [1]. While unimodal techniques are becoming more versatile [21–23], objects with a transparent material like glass still pose challenging problems. Their reflectance behavior is so different that they require an entirely different modality, such as computed tomography (CT) [24]. The transparent object must then be removed from the scene to be scanned elsewhere. In the meantime, the surrounding scene can be scanned with a more common technique. However, as the transparent object takes most of its appearance from its surroundings, it must be repositioned in the surrounding scene (physically and digitally) if we are to take reference images for comparison with rendered images. The purpose of our scene reassembly is to address this type of issue.

Our digitization technique is multimodal. Currently, such techniques seem to exist only in the context of sensor fusion [25–27]. Here, the goal is to optimize reconstruction by fusing data from different sensor modalities with complementary characteristics. Even so, the different modalities see the same object and thus work for materials with a similar reflectance behavior. The challenge is then mostly in registration of the scans. In their final remarks and suggestions for future work, Weinmann and Klein [1] discuss possible ways of combining multiple techniques tailored to different types of surface reflectance. Our pipeline is a different way to take a step in this direction.

In summary, our work makes it possible to perform multimodal digitization and scene reassembly in such a way that rendered images of the reassembled scene can be quantitatively compared to photographs of the original. This enables us to provide the first empirically founded investigation of the appearance accuracy of objects digitized using a non-optical scanner.

2. DIGITIZATION PIPELINE

We divide our pipeline into four stages: (1) geometric and photographic capture of the environment and the scene itself, (2) reconstruction of surface meshes, material BRDFs, and color space, (3) reassembly of the digital scene consisting of geometric objects, material appearance properties, and environment map, and (4) rendering and comparison with reference images. See Fig. 2 for more details. The acquisition stage requires an elaborate hardware setup. We assemble the physical scene in a black light-proof enclosure. This has five LED light tubes for scene lighting, which we capture by HDR imaging of a light probe. To acquire non-transparent geometry inside this enclosure, we use a structured light scanner consisting of a stereo camera rig and a light projector mounted on a robotic arm [28, 29]. Together with an LED based illumination arc, we also use this camera with exact control for measuring isotropic BRDFs. For transparent objects, we use a CT scanner. In the following subsections, we describe the individual steps of the pipeline with focus on details required for reproducibility and on non-standard techniques that we introduce.

A. Camera Calibration and Settings

The camera system is calibrated using standard techniques [30]. Our calibration board is an 11 by 12 black-and-white checkerboard. We take care in choosing robot camera positions that ensure a good calibration so that a small error in calibration does not cause large errors later in the pipeline. For the intrinsic calibration, we include a large variety of views to estimate good lens distortion coefficients. The same applies for stereo calibration, where both cameras must have the calibration board fully in view. To ensure low error in the extrinsic calibration, we balance good coverage of the scene and good coverage of the calibration board. Since we cannot change the camera system while collecting data, we chose a small aperture to ensure that background and projected structured light patterns are always in focus from all views. The full setup is in a dark room environment to eliminate external light, so we use a long shutter time (600 ms) to obtain sufficient exposure. A slight noise component is present in the images, but this is considered negligible. Finally, we use the estimated distortion coefficients to remove distortion from all images in the dataset so that subsequent algorithms may assume a pinhole camera model.

To avoid any compression or manipulation of the images by the camera software, in particular automatic color correction, we read the raw sensor data directly. We use bilinear interpolation to reconstruct RGB images from the raw Bayer pattern images. By doing this, we obtain a consistent RGB color space. Moreover, the raw sensor data is linear and correlates directly with radiometric quantities, which allows for better BRDF and environment map estimation in later stages of our pipeline.

We capture radiometrically relevant parts of our dataset in HDR by stacking multiple exposures [31]. More specifically, we stack 11 exposures at one-stop intervals ranging from 1 to 2048 ms. For the other parts of the dataset, we capture a single image at an exposure time of 600 ms.

B. Surface Reconstruction from Structured Light

We use a standard Gray code structured light approach to generate raw point clouds for a scene [32, 33]. With camera parameters from the calibration, we transform these point clouds into the same world coordinate system.

To reconstruct one connected triangle mesh from the point clouds, we merge them into a single point cloud and perform screened Poisson reconstruction with trimming and an octree depth of nine [34]. This technique requires point normals, so before the merging we generate normals for each point cloud as follows. We resample the point cloud down to 100,000 vertices via Poisson disk sampling [35] and then compute normals via planar fitting to a nearest neighborhood of 500 points ($\sim$16 mm radius). We then reorient all the normals according to the location of one of the cameras and transfer them back onto the original point cloud. This procedure ensures smooth continuous normals, necessary for a good performance of the mesh reconstruction algorithm. As we rely on smoothing, we cannot reconstruct features in the mesh with the same physical size as the alignment error accumulated from structured light and calibration. The aim of the chosen constants was to preserve features by striking a balance between too noisy and too smooth. The operability of the pipeline is however not sensitive to the choice of these constants.

C. Material BRDF Reconstruction

We assume that all materials in the scene, besides the glass objects, are opaque and isotropic, why we model their reflectance by BRDFs. To acquire the material properties, we combine traditional canonical gonioreflectometric sampling [36] with a BRDF interpolation (reconstruction) technique [37]. A flat sample of each non-transparent material is produced. A purposely-built light array is used to illuminate the samples from 11 unique inclinations, thus providing the surface irradiance. The array is formed as a circular arc with 11 LEDs evenly distributed from 7.5° up to 90° with 7.5° steps. One of the flat material samples is placed at the center of the circle partly traced by the light array arc. Radiance emitted by the surface is measured using the cameras mounted on the robot. The robot is used to track a path distributed over one octant of a sphere, that is from 0° to 180° azimuth and 0° to 90° elevation. The center of this sphere coincides with that of the light array and its radius is slightly larger in order to avoid collision between the robot and the array. The robot moves in steps of 7.5°. For each step, it orients the camera directly towards the center of the path and captures 11 HDR image of the sample, one for each irradiance corresponding to each of the light directions. In total, this yields 2.783 HDR images per material. Note that we do not sample at camera elevations of 90° as the viewing direction of the camera is flush with the plane of the surface, why no emitted radiance should be visible. Likewise, we do not sample at azimuths of 0° as the light array here blocks the view of sample.

In order to render a given material with its BRDFs, we need to interpolate the irradiance-radiance directions between our 2.783 sparse samples in order to fill the entire $(90 \times 90 \times 180)$ MERL-format BRDF look-up table [38]. To do so, we directly follow the reconstruction method proposed by [37]. First, we use each of the 100 BRDFs in the MERL-dataset [38] as sample points in a $90 \cdot 90 \cdot 180 = 1,458,000$ dimensional space. Then, a non-linear mapping is applied to each of the samples. The mapped samples are ordered as rows of a matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$ where m is the number of BRDF samples and d is the dimension of the space. The zero-mean matrix is computed as $\mathbf{X} - \bar{\mathbf{x}}$, with $\bar{\mathbf{x}}$ being the sample mean. From that, the Singular Value Decomposition $\mathbf{X} - \bar{\mathbf{x}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is used to compute the eigenvectors and eigenvalues of the covariance matrix of $\mathbf{X} - \bar{\mathbf{x}}$, which are given as the columns of $\mathbf{V}$ and the diagonal elements of $\mathbf{\Sigma}$, respectively. This is effectively a Principal Component Analysis (PCA) where the eigenvectors are the principal components. A matrix composed of the scaled principal components as columns are computed as $\mathbf{Q} = \mathbf{V}\mathbf{\Sigma}$.

Now, the full BRDF can be reconstructed from this Principal Component space by projection. Let $\mathbf{x}' \in \mathbb{R}^n$ be n observations of irradiance-radiance directions measured of a given material. Then, let $\bar{\mathbf{x}}' \in \mathbb{R}^n$ be the mean values and $\mathbf{Q}' \in \mathbb{R}^{n \times k}$ be the scaled eigenvectors corresponding to those n observation directions. A vector, $\mathbf{c}$, which spans the full space can be constructed by finding the linear combinations of principal components that best approximate the n observations. This can be done by solving the linear least-squares optimization problem given as:

$$\begin{aligned} \mathbf{c} &= \arg\min_{\mathbf{c}} \|(\mathbf{x}' - \bar{\mathbf{x}})' - \mathbf{Q}'\mathbf{c}\|^2 + \eta\|\mathbf{c}\|^2 \\ &= (\mathbf{Q}'^T\mathbf{Q}' + \eta\mathbf{I})^{-1}\mathbf{Q}'^T(\mathbf{x}' - \bar{\mathbf{x}}') \end{aligned} \quad \textbf{(1)}$$

Note that by adding a penalty η to the norm of c, this effectively becomes a Tikhonov Regularized Least Squares. Now, the full, mapped BRDF is reconstructed as $\mathbf{x} = \mathbf{Q}\mathbf{c} + \bar{\mathbf{x}}$. The inverse of the non-linear mapping applied to $\mathbf{X}$ is applied to $\mathbf{x}$ to get the actual, unmapped BRDF of the material. The described approach is applied to every single opaque material in the scene in order to obtain a model of their appearance.

This approach assumes that the MERL database encompasses the class of materials present in the scene. Effectively, this is a practical compromise between dense, unbiased, canonical BRDF sampling and fast, inferred, BRDF sampling. This enables us to obtain high confidence BRDFs in a matter of a few hours. The details of the approach is described by [37].

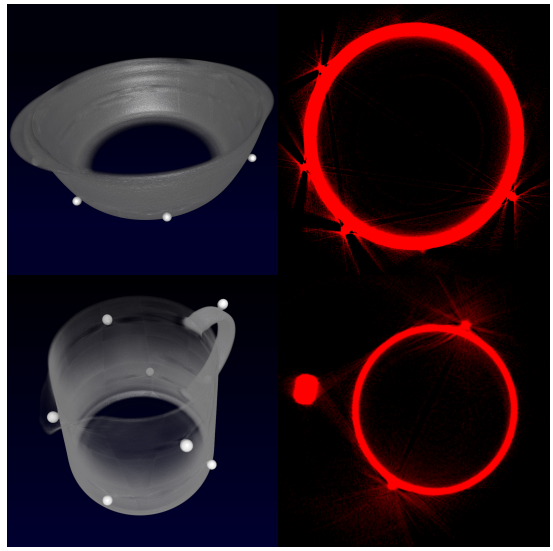
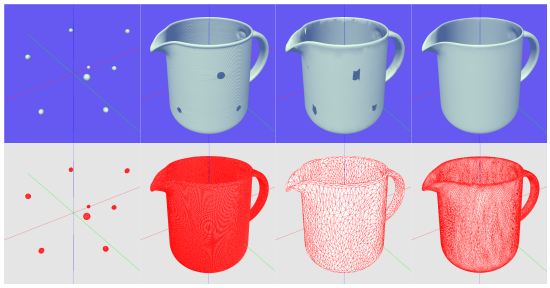


Fig. 3. CT scans of the bowl (top row) and the teapot (bottom row) with markers glued onto them. In the left column, visualized using a 1D transfer function. Note the different density of the markers. In the right column, a slice scaled to display streak artifacts.

D. Surface Reconstruction from CT

In our dataset, we have three glass objects: a sphere, a teapot (pot and lid) and a bowl (bowl and lid), for a total of five pieces. All objects have spherical plastic markers glued onto their outer surface. We CT scan each glass piece to obtain X-ray radiographs and use the CT PRO 3D reconstruction software from Nikon Metrology to obtain a volumetric image for each piece. The resolution of the reconstructed volume is up to 1000^3 voxels. Due to beam hardening, high density differences between materials lead to streak artifacts [39], especially around our markers and at the top and bottom of the objects (see Fig. 3). We account for these artifacts in the volumetric segmentation.

From a CT scan, we generate two triangular meshes with vertex normals: one for the glass object and one the plastic markers. Fig. 4 provides an overview of our procedure. We start with the markers, which appear as elements of higher density in the scan. We preprocess the scan by clamping all the values under a certain threshold to zero and then create a mesh using dual contouring [40]. Generating the glass mesh is more cumbersome. We also use dual contouring in this case, but because of the streak artifacts (Fig. 3) it is not possible to isolate the glass mesh via a threshold. Instead, we use a lower threshold



Markers | After reconstruction | After simplification | After cleaning

Fig. 4. Reconstruction from CT with stages illustrated using Phong shading (top row) and wireframe shading (bottom row). After estimating the marker mesh (first column) and fitting spheres to the markers, we reconstruct the object mesh (second column). To eliminate noise, we first simplify the mesh (third column) and then close the holes and apply our subdivision-decimation loop to get the final object mesh (fourth column).

that only removes noise, then estimate the marker positions, and use these to remove the markers from the glass mesh.

To estimate marker positions, we determine a series of center/radius pairs $(\mathbf{c}_i, r_i)$ by fitting a multi-sphere model to the marker mesh vertices using a tuned RANSAC algorithm [41]. We then carve a hole by excluding all the triangles that are inside a sphere with center $\mathbf{c}_i$ and radius $(1+\epsilon)r_i$, where ϵ is usually in the 0.5 to 0.75 range. We store the marker positions $\mathbf{c}_i$ so that we can use them to transform from the local coordinate system of the glass object to the world coordinate system (see Section F).

After removing the markers, the glass meshes still have aliasing artifacts. To deal with this issue, we first decimate the mesh down to 1% of the original vertices via quadric edge collapse. The holes are then easy to close by identifying the edge loops surrounding each hole and filling these with triangles. We then introduce a subdivision-decimation loop with alternating $\sqrt{3}$-subdivision [42] and decimation to 33% of the original vertices. We perform this subdivision-decimation operation four times to obtain a cleaned mesh. The decimation removes unwanted high frequency features from the mesh. Thus, we generate smooth meshes at the cost of some geometric precision. We are again trying to strike a balance between reconstruction error and too much smoothing. In Section 4, we compare our method with a different cleaning procedure that better preserves geometry.

E. Scene Reassembly for Non-Transparent Objects

Two operations are necessary to prepare the background mesh for rendering: labeling and deformation. In the labeling, our objective is to identify BRDFs and label each face of the mesh with a BRDF. Assuming a scene with a small number of known BRDFs, we apply edge detection and watershed on the images of the scene to segment BRDF boundaries. Shadows, specular highlights, and different viewing angles of the scene complicate fully automatic BRDF identification. Our approach gets us most of the way, but we manually correct any residual misclassification. Fig. 5 shows a label image produced by our labeling technique.

The label images can be used in multi-view projective texturing of the background mesh. However, we would like to

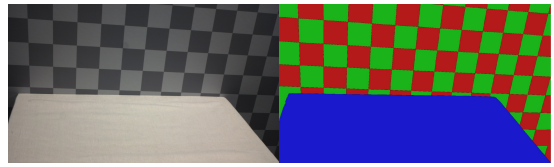


Fig. 5. Labeling of the image to the left results in the label image to the right. Each color in the label image represents a label that we assign a BRDF to. The black edges between labels indicate areas where we apply a nearest neighbor method.

No subdivision | One subdivision | Two subdivisions

Fig. 6. Subdividing the mesh dissolves unwanted boundary sawtooth artifacts that originate from the BRDF labeling.

precompute the view and label selection instead of doing it millions and millions of times while rendering. To avoid uv-unwrapping of the mesh for storing precomputed labels, we take an approach inspired by micropolygon rendering [43]. We project each vertex of a face onto the label images of the scene and select the face BRDF according to the image label that most of the face vertices were projected to. If a vertex projects to an unknown label, we resolve it by a nearest neighbor search. Since faces around material boundaries overlap multiple materials, we get sawtooth artifacts. We dissolve these by subdividing the mesh until the rendered triangles are smaller than the surface area observed in a pixel, see Fig. 6.

When applying physically based rendering, we observed intersections between background scene and glass meshes. This could be due to small errors in reconstruction and positioning, or perhaps the harder glass objects press down the tablecloth when placed for reference imaging. It causes significant visual artifacts since the rendering exposes all surfaces of a transparent object. To eliminate these artifacts, we accommodate the hard object (glass) by deforming the soft object (tablecloth), see Fig. 7 (and 16). To deform the soft object, we need a "down" direction in which to push the vertices. We first find contact vertices. These are vertices in each mesh that are close to any vertex of the other mesh. We consider vertices close if the distance between them is less than 7% of the bounding box diagonal of the hard object. Using least squares regression, we fit a contact plane to the contact vertices of the soft object. We set the sign of the contact plane normal so that the upper half-space contains the center of the hard object bounding box. Projection of a contact vertex to the normal of the contact plane then measures the

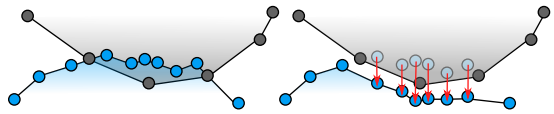


Fig. 7. Deformation of background mesh, where we push the background vertices down to avoid mesh intersection.

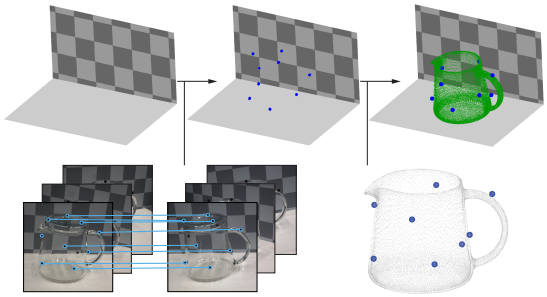


Fig. 8. Repositioning a CT scanned object in the background scene. We identify and match the markers in the stereo image pairs and calculate their corresponding 3D points. Pairing these with marker coordinates from the CT scans, we transform the CT scanned piece of an object into the world coordinate system.

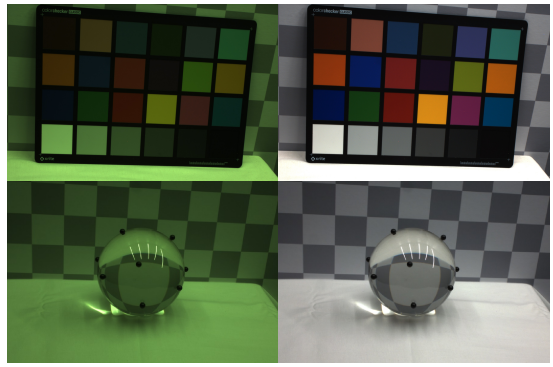


Fig. 9. Color calibration: raw images (left) and color corrected images (right). The camera sensor is particularly sensitive to green.

height of the vertex. For each soft object contact vertex $\mathbf{x}$, we find the nearest hard object contact vertices and push $\mathbf{x}$ down below the lowest one of these.

F. Scene Reassembly for Transparent Objects

To reposition the glass objects in the scene, we rigidly transform the meshes reconstructed from CT to the world coordinate system of the background mesh. We obtain this transformation by matching markers in the stereo images with the marker coordinates $\mathbf{c}_i$ computed during reconstruction from CT (see Section D).

To find the markers, we employ a size invariant circle Hough transform [44]. This works well for our dataset, where the markers show high contrast against their surroundings. We match markers in the left and the right images via Sampson distance [45]. Using this technique, markers on the same epipolar line lead to false positives, so we manually inspect the result. We also manually discard detected markers that are visible through the glass, as the refraction would lead to incorrect positioning. Markers in both stereo images with no match are discarded. The result is a set of matched markers in image coordinates as seen in Fig. 8 (bottom left). We then triangulate the matched markers from the stereo views and gather them in clusters of 3D points. We remove outliers via their distance from the cluster centers, and for each cluster we select the point with the lowest reprojection error. An example of the points and clustering is shown in Fig. 8 (top middle).

We manually pair the 3D marker coordinates from the images with the marker coordinates $\mathbf{c}_i$ from the CT scans. We perform Procrustes analysis [46] on the two point sets, excluding reflection, since we assume a rigid transformation applied to each vertex of the mesh. The bowl and the teapot are composed of multiple pieces. For these objects, we compute the transformation individually for each piece. The result of the object transformed into the scene is shown in Fig. 8 (top right). We found that in order to have low error in the transformation the chosen markers should sample the surface evenly and be visible from most views.

G. Color Calibration

Images are only quantitatively comparable if they live in the same color space. Thus, we must ensure that our radiometry-dependent data, namely reference images, environment map, and BRDFs, are in the same color space. We do this by imaging a color chart of precisely known colors. More specifically, we use second degree root-polynomial color correction [47] based on a 24 patch ColorChecker Classic from X-Rite. This provides a matrix that transforms from camera RGB to XYZ, where we assume illuminant D50 when specifying the XYZ values of the colorchecker. With the assumption of illuminant D50, we can transform colors to the CIE L*a*b* color space and then compute color difference using the ΔE_{00} metric [48]. We use this to refine our result by minimizing ΔE_{00} using the BFGS algorithm [49]. The result is in Fig. 9. The average color difference is $\Delta E_{00} = 1.97 \pm 1.21$, which is larger than 1 JND (just noticeable difference) [50], but we find it acceptable.

With a chrome bearing ball as light probe and scanned glass objects, we need the refractive indices of chrome and glass to determine reflectance, transmittance, and absorption properties. Refractive indices can be found per wavelength in tables of research papers. To use such spectral optical properties together with our trichromatic image data, we integrate them to CIE RGB using the CIE RGB color matching functions listed by Stockman and Sharpe [51]. It is important to normalize these functions [52] and to use RGB rather than XYZ [53]. This is because a refractive index is not a color, but rather a quantity that in trichromatic representation should resemble a sparse sampling of the spectrum. Thus, as recommended by other authors [54], we choose CIE RGB as our rendering color space. After transforming our image data from camera RGB to XYZ, we therefore convert them to CIE RGB [55]. As a final step, we apply Bradford chromatic adaptation [50], adapting to the originally assumed illuminant D50, so that renderings and reference images get closer to real life appearance.

In the HDR stacking of exposures, a mostly linear camera response is necessary to get a good color calibration. In addition, it is (not surprisingly) important to avoid having the colorchecker at grazing angles and to obtain accurate color reference measurements from the manufacturer.

H. Environment Lighting

To generate an environment map for the lighting of our scene, we use a method similar to the mirror probe technique [56]. We diverge from the original work by using a pinhole camera model

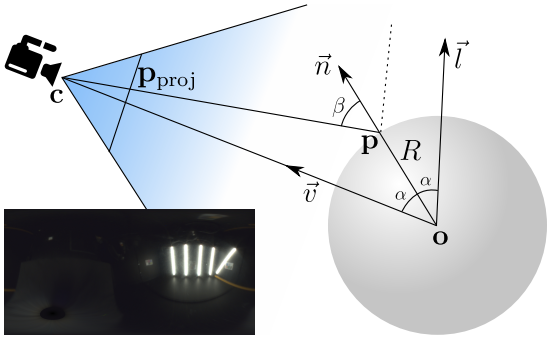


Fig. 10. Unwrapping of a spherical probe. We know the sphere radius R from specification, the camera position $\mathbf{c}$ through calibration, and the sphere center $\mathbf{o}$ by triangulation. Radiance at $\mathbf{p}_{\text{proj}}$ in our image then corresponds to the environment map direction $\vec{l}$. The result for the robot enclosure is in the lower left corner in latitude-longitude panoramic format (here tone-mapped).

instead of an orthographic one for probe image unwrapping. This is possible in our pipeline as we have a calibrated camera and know its position relative to the photographed mirror probe. With the pinhole model, we obtain more precise lighting in our renderings. The environment map is generated from HDR images and stored in latitude-longitude panoramic format [50]. We use a polished grade G100 chrome bearing ball as mirror probe.

An environment map represents an infinite area light and maps a direction to a texel. To do unwrapping, we map each texel direction $\vec{l}$ to the corresponding pixel position $\mathbf{p}_{\text{proj}}$ in a light probe image. Given the configuration illustrated in Fig. 10, we have

$$\vec{v} = \frac{\mathbf{c}-\mathbf{o}}{\|\mathbf{c}-\mathbf{o}\|},\ \vec{n} = \frac{\vec{v}+\vec{l}}{\|\vec{v}+\vec{l}\|},\ \mathbf{p} = \mathbf{o} + R\vec{n},\ \mathbf{p}_{\text{proj}} = \mathbf{M}\,[\mathbf{p}^T\ \ 1]^T,$$

where camera matrix $\mathbf{M}$ and camera position $\mathbf{c}$ are available from our calibration. The radius of the sphere R is available from the bearing ball specification, and we find the center of the sphere $\mathbf{o}$ by manually annotating the sphere and then triangulating it. We assume that the distance to the actual light along $\vec{l}$ is equal to the distance between camera and sphere $\|\mathbf{c}-\mathbf{o}\|$. This assumption works well in practice, leading to an error smaller than the uncertainty of $\mathbf{o}$ caused by the triangulation. With the original orthographic camera model, we can reconstruct the lighting for all directions except one ($-\vec{v}$). In our model, we cannot reconstruct the lighting for a set of directions ($\vec{n}\cdot\vec{v} \leq R/\|\mathbf{c}-\mathbf{o}\|$), so we set them to black. Since we do our unwrapping in world space, we can combine contributions from multiple camera views with no need to align them afterwards.

The environment map is color corrected according to Section G, which enables us to correct for the angularly dependent reflectance of chrome. The correction is to divide by Fresnel reflectance, which we compute during unwrapping. As input for Fresnel's equations, we use the angle β between $\mathbf{c}-\mathbf{p}$ and $\vec{n}$ and the complex refractive index of chrome [57] converted from spectrum to CIE RGB. The result is shown in the inset of Fig. 10.

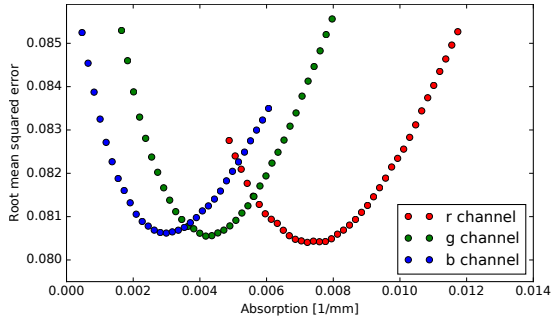


Fig. 11. Analysis by synthesis to estimate absorption of the glass bowl. We run renderings in low resolution and change the absorption in each color channel one at the time. In the case of the bowl, the blue channel is the most sensitive one.

I. Rendering

We render images using progressive unidirectional path tracing [58, 59] implemented in OptiX [60]. The captured HDR environment map is the sole light source in our scene [56]. When rendering non-specular materials, we importance sample the environment map to get direct illumination and use sampling of a cosine-weighted hemisphere to get indirect illumination. From our labeling, we have one BRDF attached to each triangle in our scene. For non-transparent objects, we use measured BRDFs tabulated in the MERL format [38]. To terminate paths probabilistically, we use Russian roulette based on the bihemispherical reflectance of each measured BRDF. This reflectance is calculated in a preprocessing step using Monte Carlo integration. We deal with transparent objects in the usual way, setting reflectance and transmittance according to Fresnel's equations of reflection and Bouguer's law of exponential attenuation. Given their small surface, we were unable to estimate a BRDF for the markers. Instead, we render them as glass with all refracted rays being absorbed.

3. ANALYSIS BY SYNTHESIS

Because we are able to render images comparable to photographs, we can use our pipeline to improve parameter estimates through analysis by synthesis. As an example, we need a scaling factor for our HDR environment map as it measures relative radiance [31]. We estimate this factor by taking ratios of references and renderings with the background scene alone. Another example is estimating real and imaginary parts of glass refractive indices. As analysis by synthesis is fundamentally ill-posed [61], we take our outset in physics-based initial guesses such as Schott K5 crown glass (sphere and teapot) and soda lime glass (bowl). Spectral refractive indices for these glasses were obtained from an online database (http://refractiveindex.info) and converted to CIE RGB. All parameters were estimated on different views than the ones shown in our comparisons of renderings with references.

An example of our analysis by synthesis is in Fig. 11, where we plot the evolution of the root-mean-squared error (RMSE) for different renderings of the glass bowl. For each rendering, we vary a trichromatic component of the absorption coefficient (which directly relates to the imaginary part of the refractive index). We identify a distinct minimum in the error for each

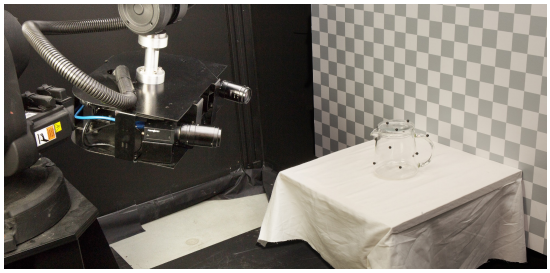


Fig. 12. Scene with checkerboard backdrop, lighting, glass teapot, and stand with table cloth observed by two cameras mounted on a 6-axis industrial robot arm.

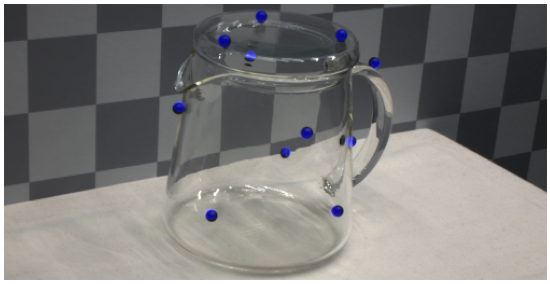


Fig. 13. Markers rendered in blue and added to the reference image to validate marker positions by looking at pixel offsets.

channel, with a slightly larger uncertainty in the red channel. The minimum values in this figure were used in our renderings of the glass bowl. We apply the same analysis to the teapot and the sphere.

Given an initial guess for a parameter, we can employ standard optimization algorithms, defining the RMSE between the reference and the rendering as a cost function to minimize. To reduce rendering times, the evaluation of the cost function can be calculated on a downsampled image or limited to a specific patch of the images. Various general optimization algorithms exist for minimizing expensive cost functions [62].

4. RESULTS

Our scenes consist of a backdrop, a stand, and a glass object (with markers) placed on the stand. The backdrop is a 30 by 20 white-and-gray checkerboard print on 120 cm by 80 cm semi-matte cardboard and the stand is a tabletop with a white cloth. An example scene is depicted in Fig. 12. We implemented our reconstruction and reassembly procedures as a modular software pipeline and computed all rendered images using our path tracer. As illustrated in Fig. 2 and mentioned in Section G, we color correct both rendered images and reference images to have a meaningful perceptual comparison. Fig. 13 compares markers in a reference image with rendered markers to validate our marker positioning. For the teapot, the average distance between the markers from stereo and the transformed markers from CT is 0.43 mm.

Fig. 14 presents pixelwise comparisons of reference images and rendered images. The error images allow us to spot subtle differences not easily noticed in a perceptual comparison, such

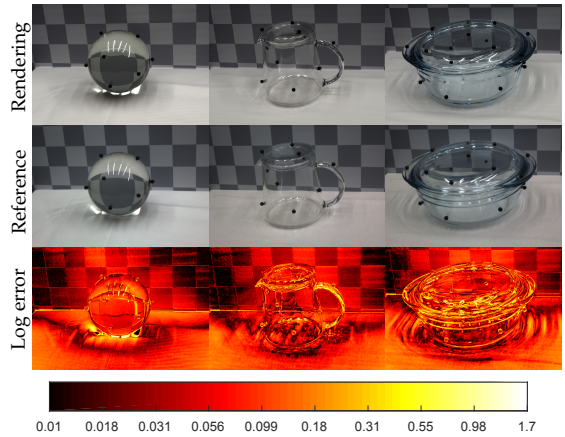


Fig. 14. Pixelwise error for three rendering-reference pairs. Error is the ℓ^2-norm of 32-bit per channel RGB images, visualized using a base 10 logarithmic scale.

as the slight misalignments in geometry and highlights. As reference photographs were not captured in HDR, we clamp the renderings correspondingly. This means that areas of strong light intensity, such as highlights and intense caustics, appear black in the error images.

Fig. 15 exemplifies the impact on error images of some of our contributions. In Fig. 15 (a), we only reposition the glass object in the background scene and apply color correction (Sections F and G). This means that we use Lambertian materials (with bihemispherical reflectances from the measured BRDFs), an orthographic unwrapping model of the environment map, and no chrome reflectance correction or analysis by synthesis optimization. We compare to the reference image in Fig. 15 (g), with error images as in Fig. 14. Fig. 15 (b) shows the impact of using measured BRDFs (Section C), resulting in a more accurate representation of the folds of the cloth in the background scene (top image) and an overall reduction of the error (bottom image). In Fig. 15 (c), we add deformation of the background mesh (Section E), which ensures that the background mesh does not poke through the glass surface (see a close-up in Fig. 16). Additionally, we can see how this improves the error on the lid of the bowl, because of refraction of light in the glass. The next step, Fig. 15 (d), shows the impact of our modified environment map unwrapping (Section H) against the standard orthographic unwrapping rotated according to our camera parameters. A close-up is available in Fig. 17. Our modified unwrapping provides a better shape and alignment of highlights and caustics. Partially due to the assumption of infinitely distant environment light, some alignment artifacts persist. In Fig 15 (e), we show the effect of correcting for chrome reflectance in our environment map reconstruction. Quantitatively, this changes the distribution of the error (bottom image). On the cloth, the exposure increases, exposing the caustics misalignment. On the backdrop, the error reduces. Interestingly, the structural similarity index (SSIM) improves while the RMSE worsens. Finally, in Fig. 15 (f), we use analysis by synthesis to adjust glass absorption. This improves the glass appearance, but it also leads to slight color changes in other parts of the scene due to indirect light paths. Because of this global influence, the analysis by synthesis introduces

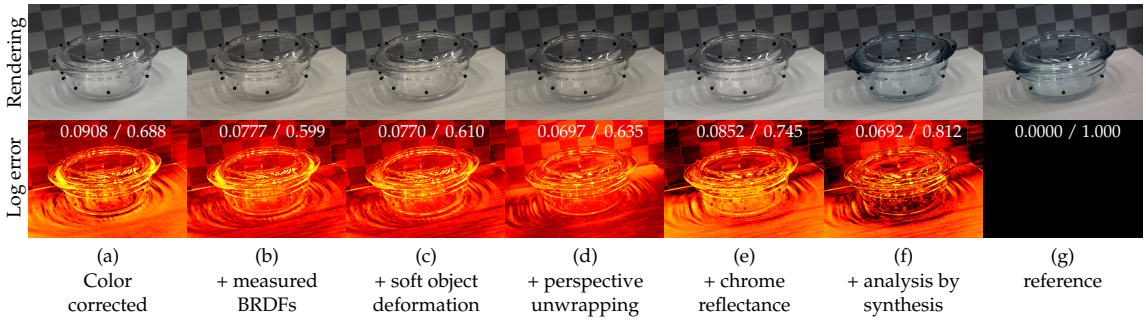


Fig. 15. Qualitative (top) and quantitative (bottom) step-by-step evaluation of our reassembly techniques. The log error images have the same format as in Fig. 14 and the reference photograph is in the rightmost column (g). In each column, we provide root-mean-squared error and structural similarity index (RMSE / SSIM). Both measures attain their best score in our final result (f).

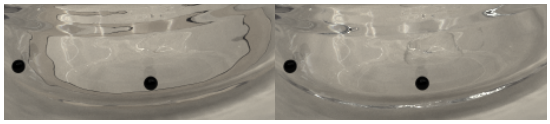


Fig. 16. Zoom-in of Figs. 15 (b) and (c) to emphasize the effect of our background deformation.

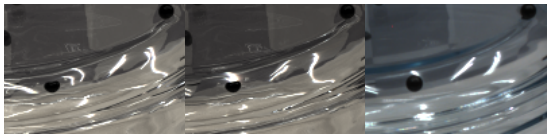


Fig. 17. Zoom-in of Fig. 15 (c) and (d) to emphasize the effect of our perspective unwrapping of the environment map.

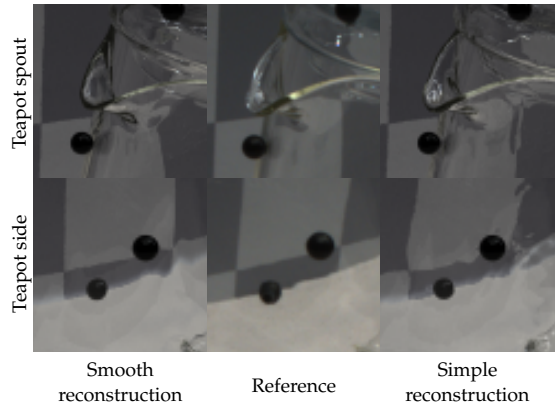


Fig. 18. Trade-off in mesh reconstruction. If we smooth more, we get less distortion in the refractions, but less precision in the mesh geometry. From left to right: Rendering with smoothing, reference image, rendering without smoothing.

slightly too much absorption to compensate for the slightly too bright tablecloth.

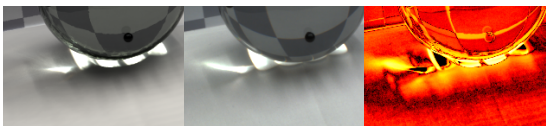
As an example of how our pipeline can be used to validate existing algorithms, we investigate the case of glass object reconstruction. In Fig. 18, we compare two different reconstruction methods with focus on two parts of the teapot scene. Smooth reconstruction refers to the procedure described in Section D. The other procedure is to simply decimate the reconstructed mesh to 2.5% of the original vertices and apply Taubin smoothing [63]. This removes the high frequencies of the noise but much noise is still present in the midranges leading to wobbly refractions. Our method in Section D reduces far more noise, but this is at the cost of greater changes to the overall shape. We note that a refractive object with a simple geometry is very hard to reconstruct automatically if fidelity and almost no noise are both required.

5. DISCUSSION

Since our pipeline enables us to compare renderings with photographs, we can identify problems in acquisition, reconstruction, and rendering that would otherwise have been hard to find. Camera calibration issues, for example, reveal themselves as error lines along edges (visible in Fig. 20). Color calibration issues reveal themselves as color shift. Such issues led us to more careful camera calibration procedures and the choice of root-polynomial color correction. Qualitative comparisons revealed artifacts in surface reconstruction, mesh intersections calling for deformation, misplacement of highlights, color shift due to chrome reflectance, and missing absorption in renderings (Figs. 15–18). Quantitative comparisons confirmed improvement due to perspective unwrapping of light probe images and led to analysis by synthesis.

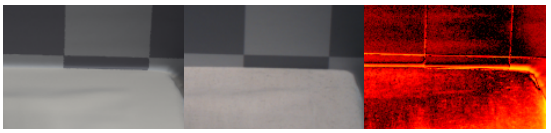
We found analysis by synthesis useful for estimating parameters with an outset in physics-based initial guesses. The results in Fig. 11 show that we can estimate optical properties for a given material and use them in a different setting (right part of Fig. 1). The precision of the estimation varies with the impact of the property on the overall error, and the estimated parameters may compensate for unrelated errors. In this regard, specific scene configurations could be used to favor estimation of a particular parameter.

The most important limitation of our method is that we describe materials as large patches of isotropic BRDFs. In our renderings, this assumptions works well for the checkerboard back-



Rendering Reference Log error (as in 14)

Fig. 19. Effect of missing subsurface scattering.



Rendering Reference Log error (as in 14)

Fig. 20. Effect of meshes connecting at the material transition: small error at the boundary between tablecloth and backdrop.

drop but not for the cloth, where we both have subsurface scattering effects and probably anisotropy due to the weave structure of the cloth. As seen in Fig. 19, we are unable to render secondary light coming through the cloth in the area next to the main caustic.

We also see a limitation at the transition between non-connected elements, here visible in the renderings at the boundary between the cloth and the backdrop (see Fig. 20). The problem derives from the fact that the cloth and the backdrop were too close to each other during dataset acquisition. This resulted in the Poisson mesh reconstruction interpreting them as a continuous object instead of two separate ones. Finally, we have artifacts around the markers from the CT reconstruction due to transition of materials. This interrupts the homogeneity in the glass and becomes visible in the renderings. Furthermore, markers are glued onto the surface of the glass, and the glue is not considered in the reconstruction and renderings. The marker glue problem is magnified by the glass refraction.

6. CONCLUSION

We have proposed a pipeline for multimodal scene digitization. Our work addresses the entire process from acquisition of the original objects, through reassembly of the digital scene, to accurate modeling of camera and environment. While the pipeline required several non-trivial steps, the benefits are correspondingly great since we can perform pixelwise comparisons between rendered images and photographs of the corresponding physical scene. This means that we have the means to quantitatively assess the accuracy of an acquired model based on comparison with empirical evidence. We believe this kind of quantitative assessment has not previously been possible for transparent objects. In applications like cultural heritage preservation and industrial inspection, where the accuracy of a digitization is important, such comparison with empirical evidence is crucial.

To the best of our knowledge, our work is also the first work to quantify the photorealism of a heterogeneous scene requiring multimodal acquisition.

Our dataset is publicly available so that others can test new techniques for the different steps of the pipeline with quantitative feedback based on photorealistic rendering. The fact that one can use off-the-shelf rendering techniques for improving the different steps of a multimodal digitization pipeline is perhaps the most important benefit of our work. An application of the full pipeline is the virtual product placement in Fig. 1. Another important application is the estimation of radiometric properties through analysis by synthesis. The ability to accurately estimate optical properties through computation rather than measurement, which might require specialized equipment, is likely to greatly simplify the digitization of radiometrically complex objects. In this paper, we estimated absorption and refractive indices of transparent objects, but analysis by synthesis could be equally useful for other materials with non-trivial BRDFs. This is another key benefit of our work that we believe is well worth exploring in the future.

Funding. Innovation Fund Denmark (IFD) (75-2014-1, 3067-00001B, 5163-00001B, 5163-00003B).

REFERENCES

1. M. Weinmann and R. Klein, "Advances in geometry and reflectance acquisition (course notes)," in "Proceedings of SIGGRAPH Asia 2015 Courses," (ACM, 2015).
2. P. Debevec, "The light stages and their applications to photoreal digitial actors," in "SIGGRAPH Asia 2012 Technical Briefs," (2012).
3. L. Gomes, O. R. P. Bellon, and L. Silva, "3D reconstruction methods for digital preservation of cultural heritage: A survey," Pattern Recognition Letters **50**, 3–14 (2014).
4. L. Zhang, H. Dong, and A. E. Saddik, "From 3D sensing to printing: A survey," ACM Transactions on Multimedia Computing, Communications, and Applications **12**, 27:1–27:23 (2015).
5. J. B. Nielsen, E. R. Eiriksson, R. L. Kristensen, J. Wilm, J. R. Frisvad, K. Conradsen, and H. Aanæs, "Quality assurance based on descriptive and parsimonious appearance models," in "Workshop on Material Appearance Modeling (MAM 2015)," (The Eurographics Association, 2015), pp. 21–24.
6. B. T. Phong, "Illumination for computer generated pictures," Communications of the ACM **18**, 311–317 (1975).
7. C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, "Modeling the interaction of light between diffuse surfaces," Computer Graphics (Proceedings of SIGGRAPH 84) **18**, 213–222 (1984).
8. A. Takagi, H. Takaoka, T. Oshima, and Y. Ogata, "Accurate rendering technique based on colorimetric conception," Computer Graphics (Proceedings of SIGGRAPH 90) **24**, 263–272 (1990).
9. G. W. Meyer, H. E. Rushmeier, M. F. Cohen, D. P. Greenberg, and K. E. Torrance, "An experimental evaluation of computer graphics imagery," ACM Transactions on Graphics **5**, 30–50 (1986).
10. H. Rushmeier, G. Ward, C. Piatko, P. Sanders, and B. Rust, "Comparing real and synthetic images: Some ideas about metrics," in "Rendering Techniques '95 (Proceedings of EGWR 1995)," (Springer, 1995), pp. 82–91.
11. K. F. Karner and M. Prantl, "A concept for evaluating the accuracy of computer generated images," in "Proceedings of Spring Conference on Computer Graphics (SCCG 1996)," (1996).
12. S. N. Pattanaik, J. A. Ferwerda, K. E. Torrance, and D. P. Greenberg, "Validation of global illumination solutions through CCD camera measurements," in "Proceedings of Color Imaging Conference (CIC 1997)," (1997), pp. 250–253.
13. N. L. Jones and C. F. Reinhart, "Parallel multiple-bounce irradiance caching," Computer Graphics Forum (Proceedings of EGSR 2016) **35**, 57–66 (2016).
14. N. L. Jones and C. F. Reinhart, "Experimental validation of ray tracing as a means of image-based visual discomfort prediction," Building and Environment **113**, 131–150 (2017).
15. D. P. Greenberg, K. E. Torrance, P. Shirley, J. Arvo, J. A.Ferwerda, S. Pattanaik, E. Lafortune, B. Walter, S.-C. Foo, and B. Trumbore, "A framework for realistic image synthesis," in "Proceedings of SIGGRAPH 97," (ACM/Addison-Wesley, 1997), pp. 477–494.
16. F. Drago and K. Myszkowski, "Validation proposal for global illumination and rendering techniques," Computers & Graphics **25**, 511–518 (2001).

17. C. Ulbricht, A. Wilkie, and W. Purgathofer, "Verification of physically based rendering algorithms," Computer Graphics Forum **25**, 237–255 (2006).
18. J. Meseth, G. Müller, R. Klein, F. Röder, and M. Arnold, "Verification of rendering quality from measured BTFs," in "Proceedings of Applied Perception in Graphics and Visualization (APGV 2006)," (ACM, 2006), pp. 127–134.
19. A. I. Ruppertsberg and M. Bloj, "Rendering complex scenes for psychophysics using RADIANCE: How accurate can you get?" Journal of the Optical Society of America A **23**, 759–768 (2006).
20. A. Dal Corso, J. R. Frisvad, T. K. Kjeldsen, and J. A. Bærentzen, "Interactive appearance prediction for cloudy beverages," in "Workshop on Material Appearance Modeling (MAM 2016)," (The Eurographics Association, 2016), pp. 1–4.
21. B. Tunwattanapong, G. Fyffe, P. Graham, J. Busch, X. Yu, A. Ghosh, and P. Debevec, "Acquiring reflectance and shape from continuous spherical harmonic illumination," ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013) **32**, 109:1–109:11 (2013).
22. T. Nöll, J. Köhler, G. Reis, and D. Stricker, "Fully automatic, omnidirectional acquisition of geometry and appearance in the context of cultural heritage preservation," Journal on Computing and Cultural Heritage **8**, Article 2 (2015).
23. H. Wu, Z. Wang, and K. Zhou, "Simultaneous localization and appearance estimation with a consumer RGB-D camera," IEEE Transactions on Visualization and Computer Graphics **22**, 2012–2023 (2016).
24. I. Ihrke, K. N. Kutulakos, H. P. A. Lensch, M. Magnor, and W. Heidrich, "Transparent and specular object reconstruction," Computer Graphics Forum **29**, 2400–2426 (2010).
25. A. Kolb, J. Zhu, and R. Yang, "Sensor fusion," in "Digital Representation of the Real World," M. A. Magnor, O. Grau, O. Sorkine-Hornung, and C. Theobalt, eds. (CRC Press, 2015), chap. 9, pp. 133–150.
26. V. Bhateja, H. Patel, A. Krishn, A. Sahu, and A. Lay-Ekuakille, "Multimodal medical image sensor fusion framework using cascade of wavelet and contourlet transform domains," IEEE Sensors Journal **15**, 6783–6790 (2015).
27. A. Pamart, O. Guillon, J.-M. Vallet, and L. De Luca, "Toward a multimodal photogrammetric acquisition and processing methodology for monitoring conservation and restoration studies," in "Eurographics Workshop on Graphics and Cultural Heritage," (The Eurographics Association, 2016), pp. 207–210.
28. H. Aanæs and A. B. Dahl, "Accuracy in robot generated image data sets," in "Proceedings of SCIA 2015," , vol. 9127 of *Lecture Notes in Computer Science* (Springer, 2015), vol. 9127 of *Lecture Notes in Computer Science*, pp. 472–479.
29. H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl, "Large-scale data for multiple-view stereopsis," International Journal of Computer Vision **120**, 153–168 (2016).
30. Z. Zhang, "A flexible new technique for camera calibration," IEEE Transactions on Pattern Analysis and Machine Intelligence **22**, 1330–1334 (2000).
31. P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in "Proceedings of SIGGRAPH 97," (ACM/Addison-Wesley, 1997), pp. 369–378.
32. J. L. Posdamer and M. Altschuler, "Surface measurement by space-encoded projected beam systems," Computer Graphics and Image Processing **18**, 1–17 (1982).
33. J. Geng, "Structured-light 3D surface imaging: a tutorial," Advances in Optics and Photonics **3**, 128–160 (2011).
34. M. Kazhdan and H. Hoppe, "Screened Poisson surface reconstruction," ACM Transactions on Graphics **32**, 29:1–29:13 (2013).
35. M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes," IEEE Transactions on Visualization and Computer Graphics **18**, 914–924 (2012).
36. J. F. Murray-Coleman and A. M. Smith, "The automated measurement of BRDFs and their application to luminaire modeling," Journal of the Illuminating Engineering Society **19**, 87–99 (1990).
37. J. B. Nielsen, H. W. Jensen, and R. Ramamoorthi, "On optimal, minimal BRDF sampling for reflectance acquisition," ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2015) **34**, 186:1–186:11 (2015).
38. W. Matusik, H. Pfister, M. Brand, and L. McMillan, "A data-driven reflectance model," ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003) **22**, 759–769 (2003).
39. J. F. Barrett and N. Keat, "Artifacts in CT: Recognition and avoidance," RadioGraphics **24**, 1679–1691 (2004).
40. T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of Hermite data," ACM Transactions Graphics (Proceedings of SIGGRAPH 2002) **21**, 339–346 (2002).
41. M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," Communications of the ACM **24**, 381–395 (1981).
42. L. Kobbelt, "$\sqrt{3}$-subdivision," in "Proceedings of SIGGRAPH 2000," (ACM/Addison-Wesley, 2000), pp. 103–112.
43. R. L. Cook, "The Reyes image rendering architecture," Computer Graphics (Proceedings of SIGGRAPH 87) **21**, 95–102 (1987).
44. T. Atherton and D. Kerbyson, "Size invariant circle detection," Image and Vision Computing **17**, 795 – 803 (1999).
45. P. D. Sampson, "Fitting conic sections to "very scattered" data: An iterative refinement of the Bookstein algorithm," Computer Graphics and Image Processing **18**, 97–108 (1982).
46. J. C. Gower, "Generalized Procrustes analysis," Psychometrika **40**, 33–51 (1975).
47. G. D. Finlayson, M. Mackiewicz, and A. Hurlbert, "Color correction using root-polynomial regression," IEEE Transactions on Image Processing **24**, 1460–1470 (2015).
48. G. Sharma, W. Wu, and E. N. Dalal, "The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations," Color Research & Application **30**, 21–30 (2005).
49. J. Nocedal and S. J. Wright, *Numerical Optimization* (Springer, 2006), 2nd ed.
50. E. Reinhard, G. Ward, S. Pattanaik, P. Debevec, W. Heidrich, and K. Myszkowski, *High Dynamic Range Imaging: Acquisition, Display and Image-Based Lighting* (Morgan Kaufmann/Elsevier, 2010), 2nd ed.
51. A. Stockman and L. T. Sharpe, "The spectral sensitivities of the middle- and long-wavelength-sensitive cones derived from measurements in observers of known genotype," Vision Research **40**, 1711–1737 (2000).
52. J. R. Frisvad, N. J. Christensen, and H. W. Jensen, "Computing the scattering properties of participating media using Lorenz-Mie theory," ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) **26**, 60:1–60:10 (2007).
53. C. Ulbricht and A. Wilkie, "A problem with the use of XYZ colour space for photorealistic rendering computations," in "Proceedings of Colour in Graphics, Imaging, and Vision (CGIV 2006)," (2006), pp. 435–437.
54. J. Meng, F. Simon, J. Hanika, and C. Dachsbacher, "Physically meaningful rendering using tristimulus colours," Computer Graphics Forum (Proceedings of EGSR 2015) **34**, 31–40 (2015).
55. H. S. Fairman, M. H. Brill, and H. Hemmendinger, "How the CIE 1931 color-matching functions were derived from Wright-Guild data," Color Research & Application **22**, 11–23 (1997).
56. P. Debevec, "Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography," in "Proceedings of SIGGRAPH 98," (ACM, 1998), pp. 189–198.
57. A. D. Rakić, A. B. Djurišić, J. M. Elazar, and M. L. Majewski, "Optical properties of metallic films for vertical-cavity optoelectronic devices," Applied optics **37**, 5271–5283 (1998).
58. J. T. Kajiya, "The rendering equation," Computer Graphics (Proceedings of SIGGRAPH 86) **20**, 143–150 (1986).
59. M. Pharr and G. Humphreys, *Physically Based Rendering: From Theory to Implementation* (Morgan Kaufmann/Elsevier, 2010), 2nd ed.
60. S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich, "OptiX: A general purpose ray tracing engine," ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010) **29**, 66:1–66:13 (2010).
61. M. Hejrati and D. Ramanan, "Analysis by synthesis: 3D object recognition by object reconstruction," in "Proceedings of IEEE Conference

on Computer Vision and Pattern Recognition (CVPR 2014)," (2014), pp. 2449–2456.
62. D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," Journal of Global Optimization **13**, 455–492 (1998).
63. G. Taubin, "A signal processing approach to fair surface design," in "Proceedings of SIGGRAPH 95," (ACM Press, 1995), pp. 351–358.

Appendix III

Interactive Directional Subsurface Scattering and Transport of Emergent Light

The Visual Computer
The final publication is available at Springer via http://dx.doi.org/10.1007/s00371-016-1207-2

Interactive Directional Subsurface Scattering and Transport of Emergent Light

Alessandro Dal Corso · Jeppe Revall Frisvad · Jesper Mosegaard · J. Andreas Bærentzen

Received: date / Accepted: date

Abstract Existing techniques for interactive rendering of deformable translucent objects can accurately compute diffuse but not directional subsurface scattering effects. It is currently common practice to gain efficiency by storing maps of transmitted irradiance. This is however not efficient if we need to store elements of irradiance from specific directions. To include changes in subsurface scattering due to changes in the direction of the incident light, we instead sample incident radiance and store scattered radiosity. This enables us to accommodate not only the common distance-based analytical models for subsurface scattering but also directional models. In addition, our method enables easy extraction of virtual point lights for transporting emergent light to the rest of the scene. Our method requires neither preprocessing nor texture parameterization of the translucent objects. To build our maps of scattered radiosity, we progressively render the model from different directions using an importance sampling pattern based on the optical properties of the material. We obtain interactive frame rates, our subsurface scattering results are close to ground truth, and our technique is the first to include interactive transport of emergent light from deformable translucent objects.

Keywords subsurface scattering · global illumination · interactive rendering · translucent objects · turbid media

A. Dal Corso (✉) · J. R. Frisvad · J. A. Bærentzen
Technical University of Denmark, Kgs. Lyngby, Denmark
E-mail: alcor@dtu.dk

J. Mosegaard
The Alexandra Institute, Aarhus, Denmark

1 Introduction

Subsurface scattering of light is a physical phenomenon that occurs in translucent materials. Milk, honey, skin, marble, and candle wax are just a few examples of translucent materials. It is possible to produce the qualitative appearance of translucency using interactive volume rendering techniques [32], but such techniques are not quantitatively accurate. With the advent of analytical models for subsurface scattering [26], it became feasible to build more accurate techniques for interactive rendering of translucent objects. The first technique of this kind [33], and more recent ones that also work for deformable objects (see Section 2), consider diffuse subsurface scattering only. In practice, this means that subsurface scattering is computed by evaluating an integral over the object surface of an analytic dipole model [26] that only depends on the distance between the points of incidence and emergence. Single scattering and other dependencies of the subsurface scattering on the direction of the incident light are neglected. Recent work in offline rendering however shows that the directional effects are not negligible [50,20,10,14].

We present an interactive technique that supports *directional* subsurface scattering without relying on precomputation or a grid for volumetric light propagation. To the best of our knowledge, our method is the first of its kind. Since the method does not rely on texture parameterization, it works for deformable and even procedurally generated geometry.

Due to reciprocity of light transport, we would ideally treat the directions of incident and emergent light equally. This is however too costly for an interactive technique. To achieve interactivity, we need caching of subsurface scattering computations. Existing techniques typically cache transmitted irradiance [25,33]

diffuse subsurface scattering | transport of emergent light | directional subsurface scattering

Fig. 1 Deforming translucent candle rendered interactively as with existing techniques (left block), with our transport of emergent light (middle block), and including directional subsurface scattering (right block). Our method is the first to support interactive rendering of the results in the right block (6 frames per second). For this scene, we use 28 scattered radiosity maps, 45 samples per direction, and 80 virtual point lights.

(total incoming light in a surface point) and use a precomputed filter to evaluate the subsurface scattering [33, 5, 29]. These techniques require that the subsurface scattering depends on distance only, whereas we need to use the direction of the incoming light. To cache another quantity, we note that subsurface scattering partly diffuses the light even if the incident light and the scattering are highly directional. Every ray of incoming light gives rise to a (non-diffuse) lobe of emergent light at all surface points. Adding up these lobes, the emergent light is in practice nearly diffuse. We therefore store scattered radiosity (outgoing light) instead of transmitted irradiance. Some of the directional subsurface scattering models also neglect dependency on the direction of emergence but still achieve improved accuracy [50, 20, 14]. Out of these, we can directly use the ones that do not rely on precomputation [20, 14].

In some existing techniques [33, 36, 4, 42], scattered radiosity is stored per vertex. To accommodate more detailed directional effects, we use more detailed maps of the scattered radiosity. We obtain these maps without requiring texture parameterization of the translucent object by rendering the object from multiple views using orthographic cameras. For each of these views, we compute a map of scattered radiosity. We can then efficiently render the translucent object from any view by look-ups into the scattered radiosity maps.

The scattered radiosity maps have two other important advantages. As long as the light source and the object are stationary, we can blend scattered radiosity maps and thereby progressively improve the rendering. Moreover, we can compute the transport of emergent light to the surrounding scene [40, 42]. To include these light paths while keeping the translucent object deformable, we generate a distribution of virtual point lights on the surface of the translucent object and set their intensity according to the scattered radiosity. These virtual point lights enable us to render the transported light using a many-light method [8]. Since we include transport of emergent light, our method is very useful for interactive rendering of scenes with the light source hidden behind a translucent object. Indirect illumination of a scene by light that has scattered through candle wax is one use case (Fig. 1). Another interesting example is light scattering through translucent lamp shades or light bulbs. To the best of our knowledge, we present the first interactive technique for transport of light emerging from deformable translucent objects.

2 Related Work

One way to obtain interactive subsurface scattering is by means of precomputation. Several early techniques rely on precomputed scattering factors that enable subsurface light transport between surface patches or from patch to vertex [33, 23, 4, 24]. These factors resemble form factors in radiosity algorithms and specify transport of transmitted irradiance to scattered radiosity. An extension of these radiosity-like techniques is to include transport of emergent light [42]. Other work is based on precomputed radiance transfer [43, 47, 49, 46], and some of this includes directional effects such as single scattering in the rendered result [43, 47, 49]. Another approach is to precompute a grid that can be used with a fast diffusion computation to render subsurface scattering in real-time [45, 48]. As opposed to our work,

all these precomputation-based methods cannot interactively render deformable translucent objects.

Some finite element methods are fast enough to enable interactive rendering of deformable translucent objects [36,34]. However, as these methods rely on diffuse incoming light (transmitted irradiance) and a multiresolution mesh (triangular or tetrahedral), they are not easily adapted for directional subsurface scattering and would typically require some mesh preprocessing.

Volume rendering techniques can quite convincingly produce the qualitative appearance of translucency at high frame rates [32,3,2,13]. While such methods are inspired by the volume rendering equation [30], they only provide a rather rough approximation of its solution. In addition, the accuracy of the subsurface scattering is limited by the resolution of the volume or the grid. Some of the more advanced methods [3,13] also propagate light using low-order spherical harmonics that effectively diffuse the subsurface scattering contribution. Other techniques, which are based on separable filtering and a depth map, also achieve real-time subsurface scattering by aiming at the qualitative appearance and sacrificing quantitative accuracy [19,18].

Fast filtering techniques can be constructed so that they approximate diffuse subsurface scattering more accurately [12,5,21,29]. The filtering is done in texture space and thus requires texture parametrization of the object surface. To avoid texture space problems, similar filtering techniques are available for light space [9] and screen space [35,27,28,29]. The performance of all these filtering techniques, however, depends heavily on the assumption that the subsurface scattering is diffuse so that the convolution kernel is only a function of the distance between the points of incidence and emergence. Our work uses light space sampling [9], but removes the assumption that subsurface scattering is diffuse. If we were to remove this assumption from texture or screen space filtering techniques and adapt them for directional subsurface scattering, they would become texture space or screen space variations of the technique that we propose. The former variation would require texture parametrization of the object surface, the latter would be view-dependent.

Another interesting approach to interactive rendering of deformable translucent objects is based on splatting [41,6]. In this approach, surface points seen from the light source are splatted as screen-aligned quads. These splats contribute according to the subsurface scattering model where they overlap surface points in the geometry buffer of the camera. On first inspection, this seems an ideal approach for interactive rendering of directional subsurface scattering. However, the directional model requires larger splats as it varies not only with distance, and it is more expensive to evaluate as tabulation is impractical. We therefore found the splatting approach too expensive.

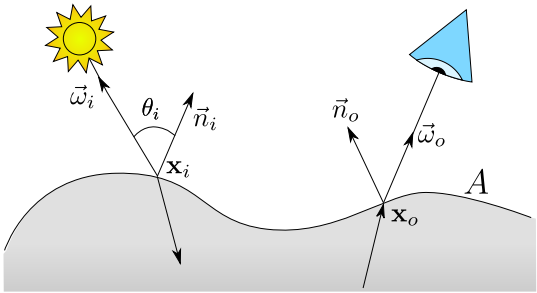


Fig. 2 BSSRDF configuration on an object surface A. The diagram illustrates the notation we use: bold font as in $\boldsymbol{x}_o$ denotes a point, while arrow overline as in $\vec{\omega}_i$ denotes a normalized direction vector.

3 Method

We render translucent objects using a *bidirectional scattering-surface reflectance distribution function* (BSSRDF). In most BSSRDFs, a translucent material is defined by the following spectral optical properties: refractive index η, absorption coefficient σ_a, scattering coefficient σ_s, and asymmetry parameter g. As is common in graphics, we use trichromatic optical properties (rgb). In addition, the BSSRDF depends on the position $\boldsymbol{x}_i$ and the direction $\vec{\omega}_i$ of the incident light as well as the position $\boldsymbol{x}_o$ and the direction $\vec{\omega}_o$ of the emergent light. The configuration is illustrated in Fig. 2. When rendering a translucent object, we obtain the outgoing radiance L_o by evaluating the following integral over all $\boldsymbol{x}_i$ in the surface area A and over all $\vec{\omega}_i$ in the hemisphere around the surface normal $\vec{n}_i$ at $\boldsymbol{x}_i$ [26]:

$$L_o(\boldsymbol{x}_o,\vec{\omega}_o) = L_e(\boldsymbol{x}_o,\vec{\omega}_o) + \int_A\int_{2\pi} S(\boldsymbol{x}_i,\vec{\omega}_i;\boldsymbol{x}_o,\vec{\omega}_o)L_i(\boldsymbol{x}_i,\vec{\omega}_i)\cos\theta_i\,\mathrm{d}\omega_i\,\mathrm{d}A_i\,, \quad (1)$$

where $\cos\theta_i = \vec{\omega}_i\cdot\vec{n}_i$, L_i is incident radiance, L_e is emitted radiance, and S is a BSSRDF. Disregarding surface reflection, as this can be incorporated using well-known techniques, the analytical BSSRDF can be written in the form:

$$S(\boldsymbol{x}_i,\vec{\omega}_i;\boldsymbol{x}_o,\vec{\omega}_o) = F_t(\vec{\omega}_o)(S_d(\boldsymbol{x}_i,\vec{\omega}_i;\boldsymbol{x}_o)+S^*)F_t(\vec{\omega}_i)\,, \quad (2)$$

where F_t is Fresnel transmittance, S_d is the diffusive part, which is typically modeled by a dipole, and S^* (dependencies omitted) is the remaining light transport, that is, the part not included with S_d.

As in other interactive subsurface scattering techniques that are not based on precomputation, we now

assume that S^* is insignificant. For most BSSRDF models [26,11,20], this means that single scattering is excluded entirely. However, if we use the directional dipole model [14], most single scattering is included with S_d. We therefore get a more accurate result with this model as the neglected S^* contains a significantly smaller part of the scattered light.

In existing interactive techniques, it is common practice to move the BSSRDF outside the integration over directions of incidence $\vec{\omega}_i$ (in Equation 1) and define transmitted irradiance by [33,9,36,35,5,41,6,34,29]

$$E(\boldsymbol{x}_i) = \int_{2\pi} L_i(\boldsymbol{x}_i, \vec{\omega}_i) F_t(\vec{\omega}_i) \cos\theta_i \, \mathrm{d}\omega_i \,. \tag{3}$$

We would however like to support BSSRDFs that include directional effects [20,14]. Since such BSSRDFs depend on $\vec{\omega}_i$, we cannot perform this separation, but we can define scattered radiosity by

$$B(\boldsymbol{x}_o) = \pi \int_A \int_{2\pi} S_d(\boldsymbol{x}_i, \vec{\omega}_i; \boldsymbol{x}_o) L_i(\boldsymbol{x}_i, \vec{\omega}_i) F_t(\vec{\omega}_i) \cos\theta_i \, \mathrm{d}\omega_i \, \mathrm{d}A_i \,. \tag{4}$$

This is an important quantity as the rendering equation (1) becomes

$$L_o(\boldsymbol{x}_o, \vec{\omega}_o) = L_e(\boldsymbol{x}_o, \vec{\omega}_o) + \frac{1}{\pi} F_t(\vec{\omega}_o) B(\boldsymbol{x}_o) \,, \tag{5}$$

which enables view-independent rendering of translucent objects if we store scattered radiosity B. We note that L_o is not fully view independent because of the Fresnel term F_t, but this is an inexpensive term that we can evaluate per pixel per frame at very little cost.

For simplicity, our initial assumption is of a scene consisting of a single object illuminated by a single directional light. In Section 3.3, we extend to point lights, and in Section 4, we show an example of using multiple lights. For surface points lit by a directional light with radiance L_ℓ and direction $\vec{\omega}_\ell$, we have

$$L_i(\boldsymbol{x}_i, \vec{\omega}_i) = L_\ell \, V(\boldsymbol{x}_i, -\vec{\omega}_\ell) \, \delta(\vec{\omega}_i + \vec{\omega}_\ell) \,, \tag{6}$$

where V is visibility and δ is a Dirac delta function that makes the inner integral disappear, yielding

$$B(\boldsymbol{x}_o) = \pi L_\ell \int_{A_{\text{lit}}} S_d(\boldsymbol{x}_i, -\vec{\omega}_\ell; \boldsymbol{x}_o) F_t(-\vec{\omega}_\ell) \cos\theta_\ell \, \mathrm{d}A_i \,, \tag{7}$$

where $\cos\theta_\ell = -\vec{\omega}_\ell \cdot \vec{n}_i$ and A_{lit} is the directly lit area of the surface (for unlit areas $L_i = V = 0$). Since we only need to integrate over the directly lit part of the surface area, we perform the integration in a geometry buffer (G-buffer) rendered from the point of view of the light source (a translucent shadow map [9]). Since we have a directional light, our G-buffer is an orthographic projection of the scene into the light's view plane, which has $\vec{\omega}_\ell$ as its normal.

In order to distribute samples in the G-buffer according to a distance r and an angle α, we assume a planar surface normal to the light direction and rewrite the integral in polar coordinates with origin $\boldsymbol{x}_o$:

$$B(\boldsymbol{x}_o) = \pi L_\ell \int_0^{2\pi} \int_0^{\infty} S_d(\boldsymbol{x}_i, -\vec{\omega}_\ell; \boldsymbol{x}_o) F_t(-\vec{\omega}_\ell) \cos\theta_\ell \, r \, \mathrm{d}r \, \mathrm{d}\alpha \,, \tag{8}$$

where $r = \|\boldsymbol{x}_o - \boldsymbol{x}_i\|$ and α is the angle between $\boldsymbol{x}_o - \boldsymbol{x}_i$ and the first basis vector of the light's view plane. This assumption is clearly often violated, but it is commonly used in derivation of BSSRDF models [26,11].

We evaluate the integral in Equation 8 by Monte Carlo integration. Our estimator for scattered radiosity is

$$B_N(\boldsymbol{x}_o) = \frac{\pi L_\ell}{N} \sum_{j=1}^{N} \frac{S_d(\boldsymbol{x}_i, -\vec{\omega}_\ell; \boldsymbol{x}_o) F_t(-\vec{\omega}_\ell) \cos\theta_\ell \, r_j}{p(r_j, \alpha_j)} \,, \tag{9}$$

where $p(r, \alpha)$ is the joint probability density function from which we draw the sample pairs (r_j, α_j). Starting from $\boldsymbol{x}_o$ transformed to the texture space of the light's camera, each sample pair corresponds to a texture space offset for looking up $\boldsymbol{x}_i$ and $\vec{n}_i$ in the light's G-buffer.

3.1 Sampling Distribution

BSSRDFs decay exponentially with the distance $r = \|\boldsymbol{x}_o - \boldsymbol{x}_i\|$. In particular, the asymptotic exponential falloff of the standard and directional dipoles [26,14] is $\exp(-\sigma_{\text{tr}} d)$, where $d \to r$ for $r \to \infty$ and σ_{tr} is the *effective transport coefficient* defined by

$$\sigma_{\text{tr}} = \sqrt{3\sigma_a(\sigma_a + (1-g)\sigma_s)} \,. \tag{10}$$

It is therefore highly beneficial to importance sample according to this exponential decay. We do importance sampling by choosing

$$p_{\text{exp}}(r, \alpha) = p(r)p(\alpha) = \sigma_{\text{tr}} e^{-\sigma_{\text{tr}} r} \frac{1}{2\pi} \,, \tag{11}$$

which is easily sampled by

$$(r_j, \alpha_j) = \left(\frac{-\log \xi_1}{\sigma_{\text{tr}}}, 2\pi\xi_2 \right) . \tag{12}$$

The symbols $\xi_1, \xi_2 \in [0, 1]$ denote canonical uniform random variables, which we obtain on the fly using a linear congruential pseudorandom number generator.

It is important to note that the effective transport coefficient σ_{tr} is different for different color bands. As a consequence, we use a separate set of position samples for each color band. In this way, we avoid color shifts, especially for materials with very different scattering coefficients in the different color bands (ketchup, for example).

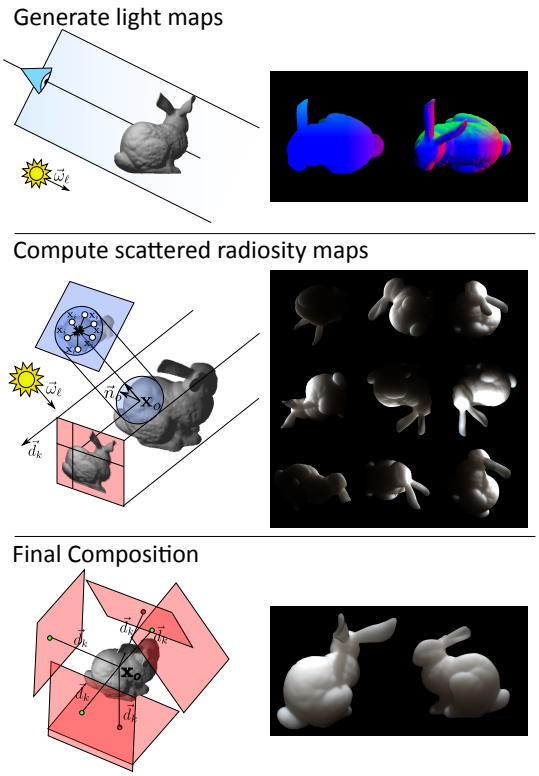


Fig. 3 Our three-step multipass technique for interactive rendering of directional subsurface scattering in deformable translucent objects. The scattered radiosity maps enable view-independence and transport of emergent light.

3.2 Rendering Technique

The diffusive part of the standard dipole BSSRDF depends only on $r = \|\boldsymbol{x}_o - \boldsymbol{x}_i\|$ and is therefore easily tabulated and used at runtime at nearly no expense. In directional subsurface scattering, on the other hand, the diffusive part of the BSSRDF depends on both $\boldsymbol{x}_o$, $\vec{n}_o$, $\boldsymbol{x}_i$, $\vec{n}_i$, and $\vec{\omega}_i$. This means that it is impractical to tabulate it and thus expensive to evaluate it. To limit the number of times that we need to evaluate the BSSRDF at runtime, we chose to exploit the opportunity to have view-independence by storing scattered radiosity in maps. In fact, as we noted in Equation 5, the scattered radiosity does not depend on the view direction $\vec{\omega}_o$. With view-independence, it is convenient to also make the update of the scattered radiosity maps progressive. By doing so, the rendered result improves over time if we are only moving the camera. Our technique is easily made progressive by adding more samples for each frame. This means that we have two render modes: (a) converged translucency with real-time fly-through and (b) fully flexible translucency rendered at interactive frame rates.

Our rendering technique is based on the rasterization pipeline of the graphics processing unit (GPU). In fully flexible mode, we use the three-step multipass algorithm illustrated in Fig. 3. In the first step, we create a G-buffer for each light source. In the second step, we compute scattered radiosity maps using these light G-buffers. In the third step, we sample the scattered radiosity maps and combine the look-ups. If nothing changed except the camera position, we also accumulate radiosity map results with the ones from the previous frames. When convergence is reached, we switch to converged mode and perform the third step only. In the following, we provide the details of the three steps.

In the first step, as in translucent shadow mapping [9], we render a G-buffer from the point of view of the light. For each pixel, we store positions and normals, as well as a material index (for global illumination purposes, Section 3.4). Each directional light has an orthographic camera and an associated G-buffer stored in a layered 2D texture. We compute all the light G-buffers in a single rendering pass, where each triangle is fed to each layer of a 2D layered texture in a geometry shader.

In the second step, we render the translucent object from K directions using orthographic cameras. The number of directions is chosen so that the surface of the model is covered well. We place the cameras randomly on the bounding sphere of the object using a quasi-random Halton sequence [22]. We then configure the cameras to look at the center of the bounding sphere with a frustum that encapsulates the sphere. Also in this step, we use layered rendering in order to efficiently render scattered radiosity into the different maps in a single pass. For each fragment of the translucent object observed by an orthographic camera, we compute the scattered radiosity by generating N samples per color band on-the-go (Equation 12), looking up into the light G-buffers with those samples to get $\boldsymbol{x}_i$ and $\vec{n}_i$, and using those to evaluate Equation 9. To avoid pattern repetition artifacts, we choose a seed for the random points using the pixel index in the scattered radiosity map as well as the current map and frame numbers.

To progressively update the scattered radiosity maps, we first perform a depth-only pass and then we render the model with writing into the depth buffer disabled. During the second step of the algorithm (except when the light condition is changing or the object is deforming), blending is enabled to allow accumulation in the scattered radiosity maps. We also generate mipmaps for the scattered radiosity maps so that we have the oppor-

Algorithm 1: Estimating the scattered radiosity in $\boldsymbol{x}_o$ using K maps (step 3 of Fig. 3). Each map has a direction $\vec{d}_k$ and a world-to-texture conversion matrix $\boldsymbol{P}_k$. The variable F counts the number of accumulated frames, which is needed to average the blending in step 2 of Fig. 3.

Data: $\boldsymbol{x}_o$, ϵ_{bias}, ϵ_{comb}, F, K
Result: B
$n = 0$
color $= (0, 0, 0)$
for $k \in [0, K)$ **do**
  $\cos\theta = \text{clamp}(\vec{n}_o \cdot \vec{d}_k, 0, 1)$
  $\bar{\boldsymbol{x}}_o = \boldsymbol{x}_o - \epsilon_{\text{comb}}(\vec{n}_o - \cos\theta\ \vec{d}_k)$
  $\bar{\boldsymbol{x}}_{o,\text{tex}} = \boldsymbol{P}_k\ \bar{\boldsymbol{x}}_o$
  $v = \texttt{multisampleVisibilityMap}_k(\bar{\boldsymbol{x}}_{o,\text{tex}}, \epsilon_{\text{bias}})$
  color $=$ **color** $+ v \cdot \texttt{sampleRadiosityMap}_k(\bar{\boldsymbol{x}}_{o,\text{tex}}, \epsilon_{\text{bias}})$
  $n = n + v$
end
$B = \frac{\textbf{color}}{Fn}$

tunity to apply a cheap high-pass filter that smoothes high frequency noise.

In the third and final pass, we sample the scattered radiosity maps for each fragment of the translucent object observed by the actual camera. This process is described in the pseudo-code in Algorithm 1. We average the contributions from the various directions with the visibility of the point as a binary weight. In the third step of Fig. 3, the green and the red dots represent the visible and not visible contributions from the point $\boldsymbol{x}_o$, respectively. Storing depth with the scattered radiosity maps, we use shadow mapping to obtain a visibility function. To avoid artifacts, we choose a constant shadow bias ϵ_{bias} for the visibility function. Moreover, to avoid errors when sampling close to the borders of a scattered radiosity map, we multi-sample the shadow map and introduce an additional bias ϵ_{comb} that translates the sample position towards the negative normal direction $-\vec{n}_o$. After composition of the scattered radiosity B, we obtain outgoing radiance from Equation 5 and perform tone mapping to finalize the result.

Considering the procedure described in this section, we can get a better understanding of the parameter N. The total number of Monte Carlo samples used for computing the outgoing radiance (L_o) in a surface point observed by the camera is $3N$ times K times the number of frames used for progressive updates. From the point of view of a surface point, N can thus be thought of as the number of samples per frame per map direction per color band.

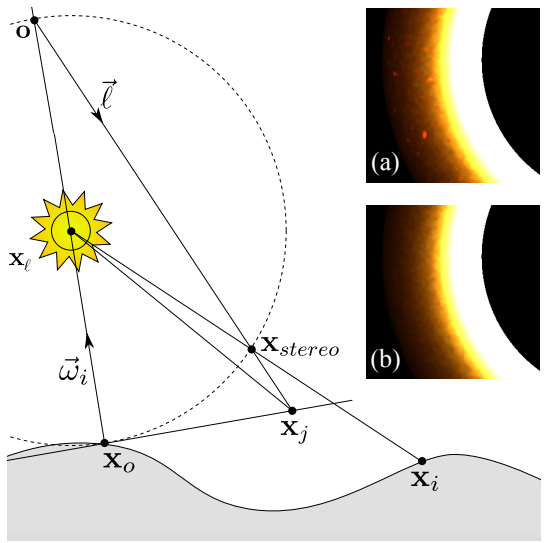


Fig. 4 Effect of stereographic correction when a translucent object surrounds a point light. With planar sampling (a), we look up into the light's cube map G-buffer using $\boldsymbol{x}_j - \boldsymbol{x}_\ell$. With stereographic correction (b), we use $\boldsymbol{x}_{\text{stereo}} - \boldsymbol{x}_\ell$ instead. The insets (a and b) show how the correction improves the final result (torus, potato material).

3.3 Point Lighting

A point light at some distance from the translucent object works much in the same way as a directional light. The light's camera simply uses perspective instead of orthographic projection and intensity falls off with the distance squared. One particularly important application of our work is however simulation of the light coming through candles, candleholders, and lamp shades (Fig. 1, for example). In these cases, the point light is surrounded by the translucent object and we then use omnidirectional shadow mapping [15] with a cube map G-buffer for the light.

With a cube map captured for a point light at $\boldsymbol{x}_\ell$, one would first get a sampled point $\boldsymbol{x}_j$ by using (r_j, α_j) to offset $\boldsymbol{x}_o$ in its tangent plane. A look-up into the cube map with $\boldsymbol{x}_j - \boldsymbol{x}_\ell$ would then provide the sampled $\boldsymbol{x}_i$ and $\vec{n}_i$. However, when observing a translucent object surrounding the light source, this planar sampling of the light's G-buffer is no longer a good approximation. To have a better approximation that enables sampling of the entire cube map for each $\boldsymbol{x}_o$ (instead of only a hemisphere), we use an inverse stereoscopic projection. With this stereoscopic correction, the direction used for look-up into the cube map becomes

$$\boldsymbol{x}_{\text{stereo}} - \boldsymbol{x}_\ell = (\boldsymbol{x}_\ell - \boldsymbol{x}_o) - 2\left[(\boldsymbol{x}_\ell - \boldsymbol{x}_o) \cdot \vec{\ell}\right] \vec{\ell}, \qquad (13)$$

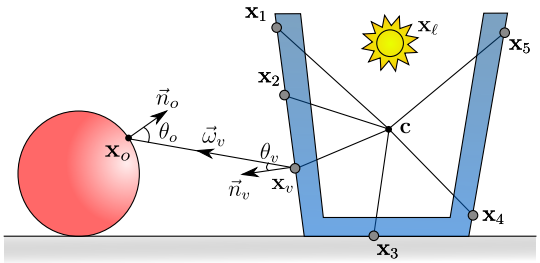


Fig. 5 Transport of emergent light from a translucent object (blue) to a diffuse object (red). We distribute VPLs (gray dots) on the outer surface of the translucent object, and use them to indirectly illuminate the remaining scene.

where

$$\vec{\ell} = \frac{(\boldsymbol{x}_j - \boldsymbol{x}_\ell) - (\boldsymbol{x}_\ell - \boldsymbol{x}_o)}{\|(\boldsymbol{x}_j - \boldsymbol{x}_\ell) - (\boldsymbol{x}_\ell - \boldsymbol{x}_o)\|}\,, \tag{14}$$

as illustrated in Fig. 4. The top right image (a) in Fig. 4 is an example of the sampling noise we get if we use $\boldsymbol{x}_j - \boldsymbol{x}_\ell$. The middle right image (b) shows how the stereoscopic correction betters this problem.

3.4 Transport of Emergent Light

We further extend our method to account for transport of emergent light using virtual point lights (VPLs) [31]. We distribute a set of N_{vpl} points on the surface of the translucent object. Then, for each observed point $\boldsymbol{x}_o$, we add the contribution from all VPLs using

$$L_o(\boldsymbol{x}_o, \vec{\omega}_o) = \sum_{v=1}^{N_{\text{vpl}}} f_r(\boldsymbol{x}_o, -\vec{\omega}_v, \vec{\omega}_o) \\ G_b(\boldsymbol{x}_o, \boldsymbol{x}_v) V(\boldsymbol{x}_o, \boldsymbol{x}_v) I_v \tag{15}$$

with VPL intensity

$$I_v = \frac{1}{\pi} F_t(\omega_v) B(\boldsymbol{x}_v) A / N_{\text{vpl}}\,, \tag{16}$$

where A is the surface area across which the VPLs were distributed, G_b is the standard bounded geometry term [8], and B is obtained from the scattered radiosity maps using Algorithm 1.

As in the previous section, we now take special steps to accommodate our key use case of a point light surrounded by a translucent material. Our approach is illustrated in Fig. 5. In this particular case, the scene illuminated by emergent light will most commonly be shadowed from surface points of the translucent object that are directly lit (as the source is surrounded). We therefore approximate the visibility term V by distributing VPLs on backlit surfaces only. With this distribution of VPLs, we use the area of the bounding volume of the translucent object as an approximation of A. This is computed for each frame on the CPU.

Unfortunately, for a deformable object and a relatively small set of VPLs, the method is prone to flickering unless we ensure that the VPL positions are stable over time. Our solution is to render the outermost surface of the translucent object to a cube map whose center $\boldsymbol{c}$ coincides with the object's bounding box center. Each pixel in the cube map now contains the coordinates of a point on the surface of the translucent object. By sampling the cube map at a constant set of random directions, we obtain a stable set of surface positions that we use as VPL locations (Fig. 5).

4 Results

The implementation of our method interactively renders directional subsurface scattering in deformable objects and requires no preprocessing nor texture parameterization of the object surface. We use the diffusive part of the directional dipole [14] as S_d or the photon beam diffusion model [20] when evaluating Equation 9. The directional dipole is significantly faster, so we use this one unless noted otherwise. We define the translucent objects in our scenes using measured optical properties from different sources [26,37,17].

To validate our results, we compare with Monte Carlo ray tracing implemented on the GPU using OptiX [38]. In this reference method, we render directional subsurface scattering using the progressive direct Monte Carlo integration technique described by Frisvad et al. [14]. As prescribed, we use a Russian roulette based on the asymptotic exponential falloff of the model to accept or reject samples. However, we do not equidistribute the samples using a dart throwing technique as a more brute force uniform sampling of the object surface is more well-suited for a GPU ray tracer. This implementation gave us a ground truth for comparison both in terms of quality and performance. However, when comparing performance, one should keep in mind that unlike our method the reference method is view dependent.

In all the following examples, performance is at interactive rates. If nothing changes except the camera, our method will converge over a number of frames and then run in real-time. The implementation switches back to interactive rates when something other than the camera changes. By 'interactive' we mean a rendering time below 166 milliseconds per frame (6 frames per second, fps), as specified by Akenine-Möller et al. [1]. All the tests were performed on an NVIDIA GeForce GTX 780 Ti graphics card (2880 cores). Unless otherwise indi-

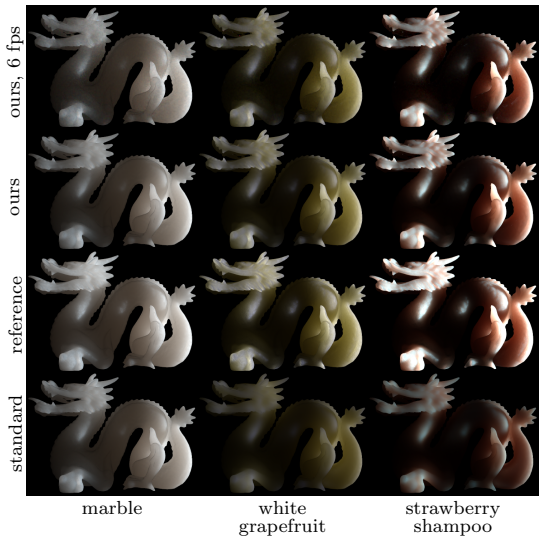


Fig. 6 Comparison of our method (rows 1 and 2) with the reference method (row 3) and diffuse subsurface scattering (row 4) for different materials. Row 1 is our results for a single frame at 6 fps, while row 2 is our view-independent result after convergence. All results use 31 maps.

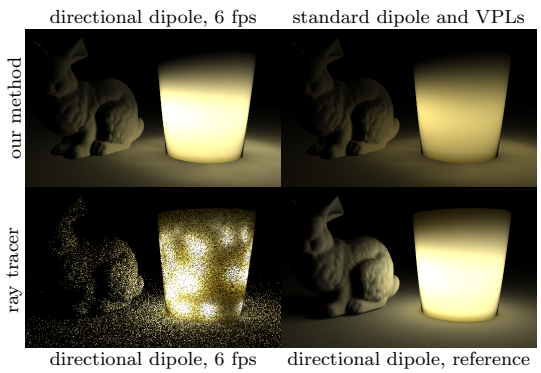


Fig. 7 Equal time comparison (left column) of our method with the reference method and qualitative comparison with diffuse subsurface scattering (upper right) and the converged reference solution (lower right). The scene is lit by a point light in a white grapefruit candle holder.

cated, our results use a 512×512 frame resolution for both radiosity and light maps.

Fig. 6 allows a visual comparison with ground truth (results obtained with the reference method). We chose one highly scattering material with isotropic phase function ($g = 0$), namely marble, and two forward scattering materials ($g > 0$), namely white grapefruit juice and strawberry shampoo. At convergence (second row), our method compares favorably to the directional dipole reference (third row). Our method improves the details

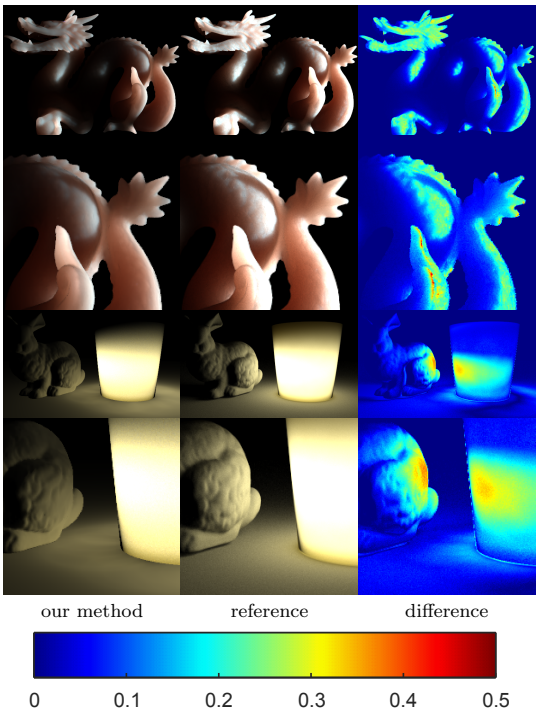


Fig. 8 Zoom-ins and differences from Figures 6 and 7. Root-mean-squared error of the color bands $\sqrt{\Delta r^2 + \Delta g^2 + \Delta b^2}$ is used as error metric in the difference images.

of the subsurface scattering when compared with diffuse subsurface scattering, that is, the standard dipole [26] (fourth row), especially for white grapefruit juice and strawberry shampoo. We also show the results of our method after one frame rendered at interactive frame rates (first row). These results are similar to our converged solution except that there is a slight bit of sampling noise, which we reduce using mipmap filtering.

Fig. 7 compares the transport of emergent light obtained with our method to that obtained with the reference method. While the 200 VPLs used here do not provide a highly accurate result, they do provide something better than a constant ambient term. At 6 fps, our solution is similar to the reference and converges very quickly to a better result, while the OptiX solution has both high-frequency and low frequency noise, is view dependent, and converges very slowly.

Fig. 8 provides zoom-ins and difference images from Figs. 6 and 7. Our results in general seem to be missing a part of the light transport. As revealed by the difference images, the missing contribution is due to undersampling of the surface at grazing incidence and missing interreflections. This undersampling is the reason why

our method ray tracer reference

Fig. 9 Stanford bunny with marble material at different scales (from top to bottom the scale is: 0.01, 0.1, and 1 meter). The left and middle columns show equal time results for our method and the ray tracer (1 frame at 6 fps). The right column shows the ray traced results after convergence. Here we use 16 maps and a 1024×1024 light map.

Mertens et al. [35] chose to sample in screen space instead of light space. However, sampling in screen space has other problems, as not all samples are lit. When considering transport of emergent light, the zoom-ins and difference images show missing shadows and inaccuracies due to the small number of VPLs. However, as graphics hardware improves, we will be able to use more VPLs and one of several fast VPL visibility techniques [8] to get better accuracy while retaining interactive frame rates.

In Fig. 9, we compare the quality reached by our solution with the quality reached by the ray traced solution in equal time. We perform this comparison for a marble bunny at three different scales. Generally, our method has a uniform behavior for different scales. For materials that are not optically thin (not at low scale), our method converges faster. The highly scattering materials (mid and high scale) are the more important cases to render well, as these are inside the range of materials for which the analytic subsurface scattering models are valid. At high scales, scattering effects become more localized, so our method is better at capturing the effect than the ray traced solution. At low scales, fewer G-buffer samples hit the object, which leads to a more noisy result with our solution.

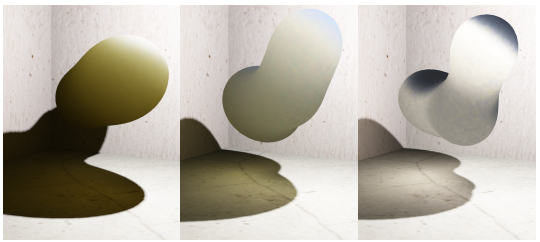


Fig. 10 Rendering with our method and a dynamically generated 3D surface ('blob') and transport of emergent light for three materials. The blob renders at 6 fps with 50 VPLs and 1500 samples per map in 6 maps. Materials from left to right: white grapefruit juice, soy milk, and glycerine soap.

In order to test the method using dynamically generated geometry, we created an implicit 3D surface [44] as the sum, $\Phi_t = \Sigma_i \phi_{i,t}(\vec{p})$, of 4 blobs,

$$\phi_{i,t}(\vec{p}) = \exp(-\sigma \|\vec{p} - \vec{p}_i(t)\|^2) \,,$$

where the position of each blob, $\vec{p}_i(t)$, is a periodic function. Since the periods are different, the period of the aggregate implicit Φ_t is potentially very large, and precomputation of the light transport inside the object would not be practical. Our method however applies, as it does not rely on precomputation, but we do need to rasterize the object. To do this, we compute a triangle mesh for an isosurface of Φ_t using dual contouring [16] implemented in a geometry shader. This is done in a pre-pass to each frame where the geometry shader evaluates Φ_t and its gradient directly based on the current time. The output triangle strips are streamed back to a vertex buffer object using transform feedback. Fig. 10 presents a rendered blob using different materials.

To justify our claimed need for scattered radiosity maps, we compare our method with an implementation without caching of subsurface scattering computations (as in translucent shadow mapping [9]). Note that this approach as opposed to ours is view dependent and pixel bound, and that unobserved VPLs would be more expensive to evaluate. Fig. 11 compares performance without considering view dependency and VPLs. Caching of scattered radiosity in maps is more efficient as soon as the translucent object occupies more than 5.8% of a 1024×1024 image.

The candle scene in Fig. 1 demonstrates the usefulness of our method. We scaled the optical properties of glycerine soap to approximate the scattering properties of candle wax. Our method creates a soft 'caustic' on the ground with varying intensity depending on the shape of the candle model. We thus enable a more realistic lighting of the scene than is obtainable with existing interactive techniques.

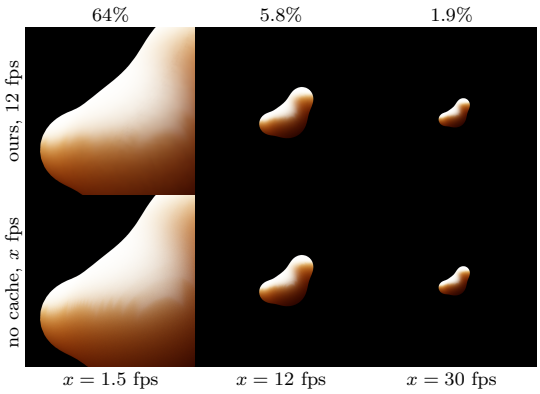


Fig. 11 Chocolate milk blob occupying different percentages of the image (noted at the top). We compare our method (ours) with a view-dependent, caching-free implementation (no cache, meaning no scattered radiosity maps). We use 1000 samples per map in 10 maps when caching, per pixel when not caching. Equal frame rates (12 fps) occur when occupancy is 5.8% of the image.

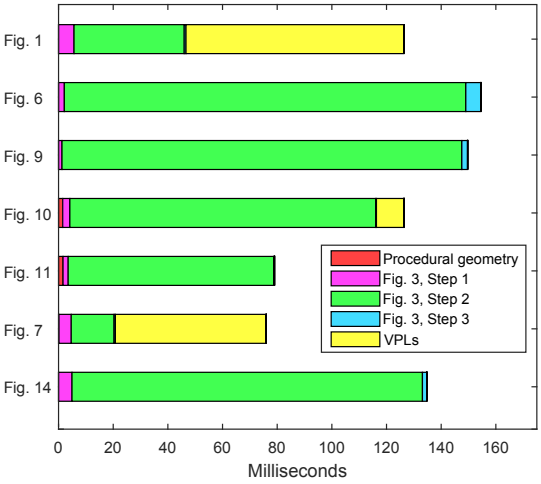


Fig. 12 Timing breakdowns for some of our renderings. Initialization times were negligible and were thus included with step 1 of Fig. 3. The evaluation of the BSSRDF and the VPLs (when present) dominate the rendering times.

To provide a performance breakdown of our technique, Fig. 12 lists render times dedicated to the different steps of our algorithm in our various results. BSSRDF evaluation (step 2 of Fig. 3) dominates all the timings, with the exception of Figs. 1 and 7, where the transport of emergent light dominates. Fig. 13 provides timings and coverage improvement of a bunny rendering with increasing K. The first seven directions cover most of the surface, while the remaining directions are necessary to cover small holes in the shading.

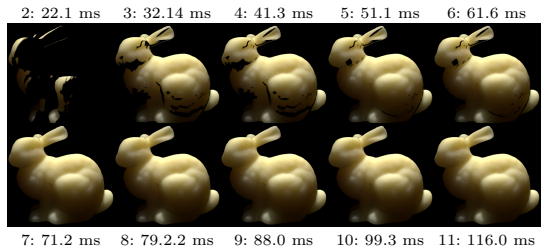


Fig. 13 Converged renderings of a potato bunny ($N = 30$) and timings for increasing number of scattered radiosity maps K. We list K followed by rendering time in milliseconds (ms) for each result. The first 7 maps cover most of the surface, while the following 4 cover small details (the small area just to the left of the bunny's hind leg, for example).

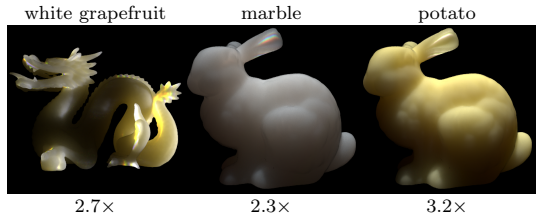


Fig. 14 Converged results with scenes and parameters as in other figures, but this time rendered using the photon beam diffusion model [20]. For each rendering, we provide the factor that this model is slower than if we use the directional dipole.

To underline the versatility of our approach, Fig. 14 has a set of results rendered using the photon beam diffusion model [20]. The weak singularities in this model lead to fireflies (overly bright pixels) with our sampling approach. We avoid this problem by clamping the distance d_r to a minimum of $0.25/(\sigma_a+\sigma_t)$ when it is used in a denominator. Factors that photon beam diffusion is slower than the directional dipole are included in the figure. These factors double if we use a graphics card with 512 cores (GTX 580) instead of 2880 cores.

Finally, Fig. 15 presents results with multiple directional lights. To approximate an environment light, we sample a number of representative directional light sources from the environment map using the method described by Pharr and Humphreys [39]. Contributions from all the directional lights are cached in the same scattered radiosity maps. In this example, we add specularly reflected light by looking up into the environment map using the direction of the reflected ray and multiplying by Fresnel reflectance.

5 Discussion

The resolution of a light's G-buffer (a light map) should be chosen carefully. If the range of the scattering ef-

Fig. 15 Stanford Bunny illuminated by an environment map. The map was importance sampled and converted to eight different directional lights. Potato material, 16 maps.

fects (roughly $1/\sigma_{\mathrm{tr}}$) is smaller than the size of one pixel in the light map, the contributions from the directional dipole tend to cluster and form 'pearling' artifacts. A possible solution would be a variation of cascaded shadow maps [51] to provide a higher resolution light map when needed. Generally, a light map of 512×512 pixels is an acceptable size that can be brought to 1024×1024 in problematic cases.

User parameters of our method include the resolutions of the light map and the scattered radiosity maps, the two biases ϵ_{comb} and ϵ_{bias}, the number of samples N, the number of scattered radiosity maps K, and the number of VPLs N_{vpl}. We now provide some guidelines for setting parameters. The size of the light map was already discussed in the previous paragraph. For the scattered radiosity maps, a size of 512×512 is fine for most application, and $K = 16$ directions generally provide enough coverage for simple models (the dragon, with its complicated geometry, required $K = 31$ directions). Performance scales linearly with K (Fig. 13), as we spend most of the time evaluating the BSSRDF (Fig. 12). The two biases ϵ_{comb} and ϵ_{bias} need to be tweaked manually. The numbers N and N_{vpl} are usually set manually to get the desired performance once the other parameters have been settled.

For most of our results, we choose the directions $\vec{d}_k$ of the scattered radiosity maps automatically. This works well for objects that are roughly convex, but for more oddly shaped concave objects some part may be left uncovered. Tearing artifacts caused by insufficient coverage appear in the mouth of the dragon in Fig. 6 and in the supplementary video. Fig. 13 also illustrates the problem, and shows that increasing the number of directions or manually choosing them can often ease this problem.

The memory consumption of our technique is comparable to that of the texture space filtering techniques [12,5,21,29]. As such, the maps and buffers that we use easily fit in the memory of modern GPUs. We surprisingly use more memory than the volumetric techniques [32,3,2,13]. The reason is that they make do with very low resolution volumes (32^3 or 64^3). It is however important to note that the added directionality and quality of details that we achieve cannot be achieved with such low resolution volumes. High resolution volumes would be needed with these techniques, which would lead to performance and memory issues.

Since we cache scattered radiosity, we cannot directly use a BSSRDF that fully depends on the direction of emergence $\vec{\omega}_o$ (the dual-beam model [10], for example). For such a BSSRDF, we would have to rely on the assumption that the emergent radiance integrates to a nearly diffuse distribution. We would then carry out a cosine-weighted integral over $\vec{\omega}_o$ when computing the scattered radiosity maps and otherwise use the same method. On the other hand, our concept of caching scattered radiosity instead of transmitted irradiance might be of interest in offline rendering techniques such as multiresolution radiosity caching [7]. This would enable use of directional subsurface scattering and inexpensive transport of emergent light in a movie production rendering solution.

6 Conclusion

We have presented a novel technique for interactive rendering of directional subsurface scattering. The method is view independent and applicable to deformable 3D models without requiring a texture parameterization of the object surface. While our method takes the direction of incident light into account, it also relies on the assumption that emergent light is not directional. This enables us to cache emergent light in so-called scattered radiosity maps. These maps enable us to control the output quality, to render progressively, and to illuminate the scene with light that has scattered through a translucent object.

Acknowledgements We would like to thank Christian Esbo Agergaard, Technical Director, Sunday Studios for the melting candle model. The Stanford Bunny and the Stanford Dragon models are courtesy of the Stanford University Computer Graphics Laboratory (http://graphics.stanford.edu/data/3Dscanrep/). The HDR environment map in Fig. 15 is courtesy of Tobias Grønbeck Andersen.

References

1. Akenine-Möller, T., Haines, E., Hoffman, N.: Real-Time Rendering, 3rd edn. A K Peters (2008)
2. Bernabei, D., Hakke-Patil, A., Banterle, F., Benedetto, M.D., Ganovelli, F., Pattanaik, S., Scopigno, R.: A parallel architecture for interactively rendering scattering and

refraction effects. IEEE Computer Graphics and Applications **32**(2), 34–43 (2012)
3. Børlum, J., Christensen, B.B., Kjeldsen, T.K., Mikkelsen, P.T., Noe, K.Ø., Rimestad, J., Mosegaard, J.: SSLPV: Subsurface light propagation volumes. In: Proceedings of ACM SIGGRAPH Symposium on High Performance Graphics (HPG '11), pp. 7–14 (2011)
4. Carr, N.A., Hall, J.D., Hart, J.C.: GPU algorithms for radiosity and subsurface scattering. In: Proceedings of Graphics Hardware 2003, pp. 51–59 (2003)
5. Chang, C.W., Lin, W.C., Ho, T.C., Huang, T.S., Chuang, J.H.: Real-time translucent rendering using GPU-based texture space importance sampling. Computer Graphics Forum (Proceedings of Eurographics 2008) **27**(2), 517–526 (2008)
6. Chen, G., Peers, P., Zhang, J., Tong, X.: Real-time rendering of deformable heterogeneous translucent objects using multiresolution splatting. The Visual Computer **28**(6–8), 701–711 (2012)
7. Christensen, P.H., Harker, G., Shade, J., Schubert, B., Batali, D.: Multiresolution radiosity caching for efficient preview and final quality global illumination in movies. Tech. Rep. Pixar Technical Memo #12-06, Pixar (2012)
8. Dachsbacher, C., Křivánek, J., Hašan, M., Arbree, A., Walter, B., Novák, J.: Scalable realistic rendering with many-light methods. Computer Graphics Forum **33**(1), 88–104 (2014)
9. Dachsbacher, C., Stamminger, M.: Translucent shadow maps. In: Proceedings of Eurographics Symposium on Rendering (EGSR 2003), pp. 197–201 (2003)
10. d'Eon, E.: A dual-beam 3D searchlight BSSRDF. In: ACM SIGGRAPH 2014 Talks, p. 65 (2014)
11. d'Eon, E., Irving, G.: A quantized-diffusion model for rendering translucent materials. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011) **30**(4), 56:1–56:13 (2011)
12. d'Eon, E., Luebke, D., Enderton, E.: Efficient rendering of human skin. In: Proceedings of Eurographics Symposium on Rendering (EGSR 2007), pp. 147–157 (2007)
13. Di Koa, M., Johan, H.: ESLPV: enhanced subsurface light propagation volumes. The Visual Computer **30**(6–8), 821–831 (2014)
14. Frisvad, J.R., Hachisuka, T., Kjeldsen, T.K.: Directional dipole model for subsurface scattering. ACM Transactions on Graphics **34**(1), 5:1–5:12 (2014)
15. Gerasimov, P.S.: Omnidirectional shadow mapping. In: R. Fernando (ed.) GPU Gems: Programming Techniques, Tips, and Tricks for Real-time Graphics, chap. 12, pp. 193–203. Addison Wesley (2004)
16. Gibson, S.F.F.: Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In: Medical Image Computing and Computer-Assisted Intervention – MICCAI'98, *Lecture Notes in Computer Science*, vol. 1496, pp. 888–898. Springer (1998)
17. Gkioulekas, I., Zhao, S., Bala, K., Zickler, T., Levin, A.: Inverse volume rendering with material dictionaries. ACM Transactions on Graphics **32**(6), 162:1–162:13 (2013)
18. Gosselin, D.R., Sander, P.V., Mitchell, J.L.: Real-time texture-space skin rendering. In: W. Engel (ed.) ShaderX^3: Advanced Rendering with DirectX and OpenGL, chap. 2.8, pp. 171–184. Charles River Media (2004)
19. Green, S.: Real-time approximations to subsurface scattering. In: R. Fernando (ed.) GPU Gems: Programming Techniques, Tips, and Tricks for Real-time Graphics, chap. 16, pp. 263–278. Addison Wesley (2004)
20. Habel, R., Christensen, P.H., Jarosz, W.: Photon beam diffusion: A hybrid Monte Carlo method for subsurface scattering. Computer Graphics Forum (Proceedings of EGSR 2013) **32**(4), 27–37 (2013)
21. Hable, J., Borshakov, G., Heil, J.: Fast skin shading. In: W. Engel (ed.) ShaderX^7: Advanced Rendering Techniques, chap. 2.4, pp. 161–173. Charles River Media (2009)
22. Halton, J.H.: Algorithm 247: Radical-inverse quasi-random point sequence. Communications of the ACM **7**(12), 701–702 (1964)
23. Hao, X., Baby, T., Varshney, A.: Interactive subsurface scattering for translucent meshes. In: Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics (i3D 2003), pp. 75–82 (2003)
24. Hao, X., Varshney, A.: Real-time rendering of translucent meshes. ACM Transactions on Graphics **23**(2), 120–142 (2004)
25. Jensen, H.W., Buhler, J.: A rapid hierarchical rendering technique for translucent materials. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) **21**(3), 576–581 (2002)
26. Jensen, H.W., Marschner, S.R., Levoy, M., Hanrahan, P.: A practical model for subsurface light transport. In: Proceedings of ACM SIGGRAPH 2001, pp. 511–518 (2001)
27. Jimenez, J., Sundstedt, V., Gutierrez, D.: Screen-space perceptual rendering of human skin. ACM Transactions on Applied Perception **6**(4), 23:1–23:15 (2009)
28. Jimenez, J., Whelan, D., Sundstedt, V., Gutierrez, D.: Real-time realistic skin translucency. IEEE Computer Graphics and Applications **30**(4), 32–41 (2010)
29. Jimenez, J., Zsolnai, K., Jarabo, A., Freude, C., Auzinger, T., Wu, X.C., von der Pahlen, J., Wimmer, M., Gutierrez, D.: Separable subsurface scattering. Computer Graphics Forum (2015). To appear
30. Kajiya, J.T., Von Herzen, B.P.: Ray tracing volume densities. Computer Graphics (Proceedings of ACM SIGGRAPH 84) **18**(3), 165–174 (1984)
31. Keller, A.: Instant radiosity. In: Proceedings of ACM SIGGRAPH 97, pp. 49–56 (1997)
32. Kniss, J., Premože, S., Hansen, C., Ebert, D.: Interactive translucent volume rendering and procedural modeling. In: Proceedings of IEEE Visualization 2002, pp. 109–116 (2002)
33. Lensch, H.P.A., Goesele, M., Bekaert, P., Kautz, J., Magnor, M.A., Lang, J., Seidel, H.P.: Interactive rendering of translucent objects. In: Proceedings of Pacific Graphics (PG 2002), pp. 214–224 (2002)
34. Li, D., Sun, X., Ren, Z., Lin, S., Tong, Y., Guo, B., Zhou, K.: TransCut: Interactive rendering of translucent cutouts. IEEE Transactions on Visualization and Computer Graphics **19**(3), 484–494 (2013)
35. Mertens, T., Kautz, J., Bekaert, P., Reeth, F.V., Seidel, H.P.: Efficient rendering of local subsurface scattering. In: Proceedings of Pacific Graphics (PG 2003), pp. 51–58 (2003)
36. Mertens, T., Kautz, J., Bekaert, P., Seidel, H.P., Reeth, F.V.: Interactive rendering of translucent deformable objects. In: Proceedings of Eurographics Symposium on Rendering (EGSR 2003), pp. 130–140 (2003)
37. Narasimhan, S.G., Gupta, M., Donner, C., Ramamoorthi, R., Nayar, S.K., Jensen, H.W.: Acquiring scattering properties of participating media by dilution. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2006) **25**(3), 1003–1012 (2006)

$$2C_1 \approx \begin{cases} 0.919317 - 3.4793\eta + 6.75335\eta^2 - 7.80989\eta^3 \\ \qquad + 4.98554\eta^4 - 1.36881\eta^5, \eta < 1 \\ -9.23372 + 22.2272\eta - 20.9292\eta^2 + 10.2291\eta^3 \\ \qquad - 2.54396\eta^4 + 0.254913\eta^5, \eta \geq 1 \end{cases} \quad 3C_2 \approx \begin{cases} 0.828421 - 2.62051\eta + 3.36231\eta^2 - 1.95284\eta^3 \\ \qquad + 0.236494\eta^4 + 0.145787\eta^5, \eta < 1 \\ -1641.1 + \frac{135.926}{\eta^3} - \frac{656.175}{\eta^2} + \frac{1376.53}{\eta} + 1213.67\eta \\ -568.556\eta^2 + 164.798\eta^3 - 27.0181\eta^4 + 1.91826\eta^5, \eta \geq 1 \end{cases}$$

Fig. 16 Approximate fits for $2C_1$ and $3C_2$ by d'Eon and Irving [11].

38. Parker, S.G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., Stich, M.: OptiX: a general purpose ray tracing engine. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2010) **29**(4), 66:1–66:13 (2010)
39. Pharr, M., Humphreys, G.: Physically Based Rendering: From Theory to Implementation, second edn. Morgan Kaufmann/Elsevier (2010)
40. Rushmeier, H.E., Torrance, K.E.: Extending the radiosity method to include specularly reflecting and translucent materials. ACM Transactions on Graphics **9**(1), 1–27 (1990)
41. Shah, M.A., Konttinen, J., Pattanaik, S.: Image-space subsurface scattering for interactive rendering of deformable translucent objects. IEEE Computer Graphics and Applications **29**(1), 66–78 (2009)
42. Sheng, Y., Shi, Y., Wang, L., Narasimhan, S.G.: A practical analytic model for the radiosity of translucent scenes. In: Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D 2013), pp. 63–70 (2013)
43. Sloan, P.P., Hall, J., Hart, J., Snyder, J.: Clustered principal components for precomputed radiance transfer. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) **22**(3), 382–391 (2003)
44. Velho, L., Gomes, J., de Figueiredo, L.H.: Implicit objects in computer graphics. Springer (2002)
45. Wang, J., Zhao, S., Tong, X., Lin, S., Lin, Z., Dong, Y., Guo, B., Shum, H.Y.: Modeling and rendering of heterogeneous translucent materials using the diffusion equation. ACM Transactions on Graphics **27**(1), 9:1–9:18 (2008)
46. Wang, R., Cheslack-Postava, E., Wang, R., Luebke, D., Chen, Q., Hua, W., Peng, Q., Bao, H.: Real-time editing and relighting of homogeneous translucent materials. The Visual Computer **24**(7), 565–575 (2008)
47. Wang, R., Tran, J., Luebke, D.: All-frequency interactive relighting of translucent objects with single and multiple scattering. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) **24**(3), 1202–1207 (2005)
48. Wang, Y., Wang, J., Holzschuch, N., Subr, K., Yong, J.H., Guo, B.: Real-time rendering of heterogeneous translucent objects with arbitrary shapes. Computer Graphics Forum (Proceedings of Eurographics 2010) **29**(2), 497–506 (2010)
49. Xu, K., Gao, Y., Li, Y., Ju, T., Hu, S.M.: Real-time homogenous translucent material editing. Computer Graphics Forum **26**(3), 545–552 (2007)
50. Yan, L.Q., Zhou, Y., Xu, K., Wang, R.: Accurate translucent material rendering under spherical Gaussian lights. Computer Graphics Forum **31**(7), 2267–2276 (2012)
51. Zhang, F., Sun, H., Nyman, O.: Parallel-split shadow maps on programmable GPUs. In: GPU Gems 3, chap. 10. Addison-Wesley (2007)

A The Directional Dipole Model

[This appendix is not in the final publication.]

As other BSSRDF models, the directional dipole uses a number of inputs that are based on the optical properties of the translucent material (η, σ_s, σ_a, g):

$$\begin{array}{llll} \sigma_t = \sigma_s + \sigma_a\,, & \sigma_s' = (1-g)\sigma_s\,, & \sigma_t' = \sigma_s' + \sigma_a\,, \\ D = 1/(3\sigma_t')\,, & d_e = 2.131\,D\sqrt{\sigma_t'/\sigma_s'}\,, & \sigma_{\mathrm{tr}} = \sqrt{\sigma_a/D}\,, \\ A = \frac{1-C_{\boldsymbol{E}}}{2C_\phi}\,, & C_\phi = \frac{1}{4}(1-2C_1)\,, & C_{\boldsymbol{E}} = \frac{1}{2}(1-3C_2)\,, \end{array}$$

where C_1 and C_2 are functions of η listed in Fig. 16. The directional dipole formulas for S_d are [14]:

$$S_d(\boldsymbol{x}_i, \vec{\omega}_i; \boldsymbol{x}_o) = S_d'(\boldsymbol{x}_o - \boldsymbol{x}_i, \vec{\omega}_{12}, d_r) - S_d'(\boldsymbol{x}_o - \boldsymbol{x}_v, \vec{\omega}_v, d_v)$$

and

$$\begin{aligned} &S_d'(\boldsymbol{x}, \vec{\omega}_{12}, r) = \\ &\tfrac{1}{4C_\phi(1/\eta)}\tfrac{1}{4\pi^2}\tfrac{e^{-\sigma_{\mathrm{tr}}r}}{r^3}\Bigg[C_\phi(\eta)\left(\tfrac{r^2}{D} + 3(1+\sigma_{\mathrm{tr}}r)\,\boldsymbol{x}\cdot\vec{\omega}_{12}\right) \\ &- C_{\boldsymbol{E}}(\eta)\Bigg(3D(1+\sigma_{\mathrm{tr}}r)\,\vec{\omega}_{12}\cdot\vec{n}_o \\ &- \left((1+\sigma_{\mathrm{tr}}r) + 3D\tfrac{3(1+\sigma_{\mathrm{tr}}r)+(\sigma_{\mathrm{tr}}r)^2}{r^2}\boldsymbol{x}\cdot\vec{\omega}_{12}\right)\boldsymbol{x}\cdot\vec{n}_o\Bigg)\Bigg], \end{aligned}$$

where the various S_d' arguments are based on positions ($\boldsymbol{x}_i$, $\boldsymbol{x}_o$), direction of incidence ($\vec{\omega}_i$), and normals ($\vec{n}_i$, $\vec{n}_o$). For the real source, we have

$$\begin{aligned} \vec{\omega}_{12} &= \frac{1}{\eta}((\vec{\omega}_i\cdot\vec{n}_i)\vec{n}_i - \vec{\omega}_i) - \vec{n}_i\sqrt{1-\frac{1}{\eta^2}(1-(\vec{\omega}_i\cdot\vec{n}_i)^2)} \\ d_r^2 &= \begin{cases} |\boldsymbol{x}_o-\boldsymbol{x}_i|^2 + D\mu_0(D\mu_0 - 2d_e\cos\beta), & \text{for } \mu_0 > 0 \\ |\boldsymbol{x}_o-\boldsymbol{x}_i|^2 + 1/(3\sigma_t)^2 & \text{, otherwise} \end{cases} \\ \mu_0 &= -\vec{n}_o\cdot\vec{\omega}_{12} \\ \cos\beta &= -\sqrt{\frac{|\boldsymbol{x}_o-\boldsymbol{x}_i|^2 - (\boldsymbol{x}\cdot\vec{\omega}_{12})^2}{|\boldsymbol{x}_o-\boldsymbol{x}_i|^2 + d_e^2}}\,, \end{aligned}$$

and for the virtual source,

$$\begin{aligned} \boldsymbol{x}_v &= \boldsymbol{x}_i + 2Ad_e\vec{n}_i^* \\ \vec{\omega}_v &= \vec{\omega}_{12} - 2(\vec{\omega}_{12}\cdot\vec{n}_i^*)\vec{n}_i^* \\ \vec{n}_i^* &= \begin{cases} \vec{n}_i & \text{, for } \boldsymbol{x}_o = \boldsymbol{x}_i \\ \dfrac{\boldsymbol{x}_o-\boldsymbol{x}_i}{|\boldsymbol{x}_o-\boldsymbol{x}_i|} \times \dfrac{\vec{n}_i\times(\boldsymbol{x}_o-\boldsymbol{x}_i)}{|\vec{n}_i\times(\boldsymbol{x}_o-\boldsymbol{x}_i)|} & \text{, otherwise.} \end{cases} \end{aligned}$$

Appendix IV

Interactive Stable Ray Tracing

Interactive Stable Ray Tracing

Alessandro Dal Corso
NVIDIA
Technical University of Denmark
alcor@dtu.dk

Marco Salvi
NVIDIA

Craig Kolb
NVIDIA

Jeppe Revall Frisvad
Technical University of Denmark

Aaron Lefohn
NVIDIA

David Luebke
NVIDIA

ABSTRACT

Interactive ray tracing applications running on commodity hardware can suffer from objectionable temporal artifacts due to a low sample count. We introduce stable ray tracing, a technique that improves temporal stability without the over-blurring and ghosting artifacts typical of temporal post-processing filters. Our technique is based on sample reprojection and explicit hole filling, rather than relying on hole-filling heuristics that can compromise image quality. We make reprojection practical in an interactive ray tracing context through the use of a super-resolution bitmask to estimate screen space sample density. We show significantly improved temporal stability as compared with supersampling and an existing reprojection techniques. We also investigate the performance and image quality differences between our technique and temporal antialiasing, which typically incurs a significant amount of blur. Finally, we demonstrate the benefits of stable ray tracing by combining it with progressive path tracing of indirect illumination.

CCS CONCEPTS

•**Computing methodologies** →**Ray tracing;**

KEYWORDS

Reprojection, dynamic scene, caching, temporal stability, GPU

ACM Reference format:
Alessandro Dal Corso, Marco Salvi, Craig Kolb, Jeppe Revall Frisvad, Aaron Lefohn, and David Luebke. 2017. Interactive Stable Ray Tracing. In *Proceedings of HPG '17, Los Angeles, CA, USA, July 28-30, 2017,* 10 pages.
DOI: 10.1145/3105762.3105769

1 INTRODUCTION

A rendered image will contain aliasing artifacts in regions where the underlying signal carries higher frequency content than the local sampling rate can capture. For example, light reflected from a highly specular surface can lead to aliasing if not sampled at sufficiently high rate. In addition, such aliasing artifacts will be perceived as particularly objectionable if high-frequency details are inconsistently sampled, causing sample values to change rapidly in time. To eliminate these artifacts, the underlying signal should ideally be bandlimited to remove frequencies beyond the local Nyquist limit. In general, however, robustly bandlimiting reflectance functions, visibility, and programmable shaders are open problems.

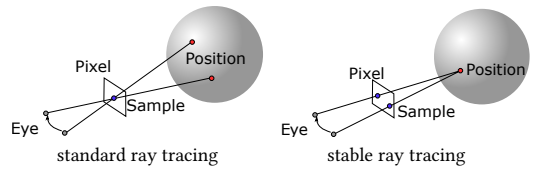


Figure 1: In standard ray tracing, sampled screen space locations are kept fixed and shading locations vary. In stable ray tracing, shading locations are kept static while screen space locations vary.

Stable shading is one strategy that can successfully mitigate aliasing artifacts in practice. In stable shading, shading calculations are performed in an object-local parametrization space, such as at the vertices of an underlying mesh, and the resulting values are interpolated across image pixels. These same object-local vertices are typically shaded again in subsequent frames, improving temporal stability in the presence of aliasing. Gouraud shading [1971] and the REYES rendering algorithm [1987], for example, use this approach to improve temporal image quality. However, these stable shading techniques do not work well with approaches such as ray tracing, wherein shading locations are determined independently of and without regard to any underlying local surface parametrization.

Stable ray tracing is a general technique that draws inspiration from previous stable shading approaches to improve the visual quality and/or reduce the computational cost of generating a sequence of images using ray tracing. Rather than using independent rays to sample the screen, shading locations from previous frames are re-used when possible, as shown in Figure 1. The fact that the points being shaded are temporally coherent results in fewer objectionable artifacts, even though the resulting images are still aliased. Furthermore, intermediate shading values can be cached along with the shading location, providing an additional performance benefit.

Our stable ray tracing is based on sample reprojection [Adelson and Hodges 1995; Badt 1988; Martin et al. 2002]. The main challenges in reprojection are verifying visibility of reprojected samples and avoiding large holes in the resulting screen space sampling pattern. We deal with the first issue by tracing visibility rays from the camera to the reprojected samples. For the second issue, we generate new samples on demand, where the demand is determined

HPG '17, Los Angeles, CA, USA

DOI: 10.1145/3105762.3105769

using screen space sample density estimation. We perform this density estimation efficiently using a super-resolution bitmask that maps subpixel sample locations. This bitmask is also useful for removing samples to keep a uniform sample distribution. As an example application of stable ray tracing, we use amortized sampling to add progressively path traced indirect illumination to an image. We demonstrate how our stable ray tracing significantly improves temporal stability as compared with supersampling and as compared with an existing reprojection technique [Martin et al. 2002]. In addition, we use an image sharpness metric to verify that our technique avoids the blur of post-process filtering techniques.

2 RELATED WORK

The use of sample reprojection to exploit the temporal coherence of ray traced frames was first suggested by Badt [Badt 1988]. His technique is limited to viewpoint changes only, but he identifies the key issues of bad pixels and missed pixels. Bad pixels occur in regions where the colors of reprojected samples are no longer valid. Missed pixels are pixels that are not hit by reprojected samples. Badt suggests the interesting notion of a "recast mat", a one-bit-per-pixel mask pointing out the pixels for which we need new samples. We reverse this concept and use a super-resolution bitmask pointing out the subpixels that were hit by a reprojected sample.

Chapman et al. [1991] map out the spatio-temporal coherence of a predefined animation sequence by tracking sample trajectories across scene geometry. This is similar to sample reprojection and also works for moving objects. A reprojection based technique exploiting coherence between frames in a predefined animation sequence is also available for variance reduction in Monte Carlo ray tracing [Zhou and Chen 2015]. Gröller and Purgathofer [1991] present a spatial data structure for techniques like these that assume a predefined animation sequence. A more progressive approach is however required in interactive ray tracing, where the future scene dynamics are unknown. Murakami and Hirota [1992] present such an incremental approach, but only for a fixed viewpoint. They connect ray paths with objects using a hash index so that it is only necessary to recompute paths that interact with dynamic objects. We also connect samples to objects using an index.

Adelson and Hodges [1995] present a fully general reprojection technique for ray tracing with a screen space data structure containing one sample per pixel. We enhance this data structure by enabling a nonintegral number of samples per pixel. Adelson and Hodges [1995] also provide a careful description of the verification phase including the need for shadow and visibility rays to check for occlusion. We adopt their verification phase and make it practical for an interactive ray tracer running on graphics hardware.

The render cache concept [Walter et al. 2002, 1999; Zhu et al. 2005] achieves interactive frame rates through reprojection with different heuristics for handling bad and missed pixels. While the heuristics significantly improve performance, they also lead to objectionable visual artifacts.

Although reprojection started out as a way of exploiting temporal coherence to save computations, Martin et al. [2002] recognize it as an important technique for avoiding temporal aliasing. They find that reprojection achieves temporal stability similar to supersampling at a significantly lower computational cost. Their system only accounts for viewpoint changes and they apply temporal filtering using a box filter spanning three frames. Apart from this, their technique seems quite similar to that of Adelson and Hodges [1995]. Martin et al. [2002] also use one sample per pixel and pick the closest sample when multiple samples land in one pixel. This one-sample-per-pixel policy easily leads to scintillation artifacts due to insertion or removal of samples as objects rotate or move relative to the camera. Missed pixels and multiple samples in one pixel occur frequently when samples move across pixel boundaries (especially in perspective view) even if the local sample density is not changing much. We successfully mitigate this issue by estimating sample density in a 2-by-2 pixels area centered in every pixel. Our super-resolution bitmask strategy enables us to perform this density estimation efficiently.

In rasterization, the use of reprojection seems to be introduced in the context of warping one rendered image to the next [Chen and Williams 1993; Mark et al. 1997]. Rasterization-based techniques like the edge and point image [Bala et al. 2003; Velázquez-Armendáriz et al. 2006] achieve good results by adding edge information to the render cache information. However, this requires precomputation of an edge-based data structure [2003] or an additional edge rendering of the image [2006]. This becomes expensive in geometry-rich scenes where several edges may land in a pixel.

Inspired by the offline techniques [Adelson and Hodges 1995; Walter et al. 1999], reprojection finds an efficient implementation in a rasterization context with the reverse reprojection cache [Nehab et al. 2007] (also discovered by Scherzer et al. [2007] in a shadow mapping context and optimized by Sitthi-amorn et al. [2008a,b]). We keep forward reprojection, as this is better suited for ray tracing. As an add-on, these techniques [Nehab et al. 2007; Scherzer et al. 2007] introduce amortized sampling where pixel values are progressively updated over time. We use such amortized sampling for progressive sampling of indirect illumination.

Reprojection has also been used together with Monte Carlo ray tracing techniques like bidirectional path tracing and photon mapping [Havran et al. 2003; Tawara et al. 2004]. These techniques rely on stored sample points in any case, so no additional data structure is needed for the reprojection. In our case, we add a screen space data structure to support stable ray tracing. Our approach is thus well-suited for unidirectional Monte Carlo techniques.

In rasterization, Herzog et al. [2010] find that temporal finite differences are useful for amortized upsampling of images rendered with real-time global illumination techniques. They investigate screen-space ambient occlusion and indirect illumination from virtual point lights. In addition to better performance, they also find that their reprojection cache improves temporal stability.

On the side of temporal stability, recently introduced postprocessing filters like temporal supersampling [Karis 2014; Patney et al. 2016] efficiently hide temporal aliasing at the cost of introducing blur in the final image. Reprojection helps avoid excessive blurring and is effective in combination with sampling and filtering techniques from antialiasing [Jimenez et al. 2012] and from denoising [Iglesias-Guitian et al. 2016]. We set out to confirm that forward reprojection also has this ability to reduce temporal aliasing while preserving image sharpness. In addition, we exemplify the benefits of having stable samples in interactive ray tracing.

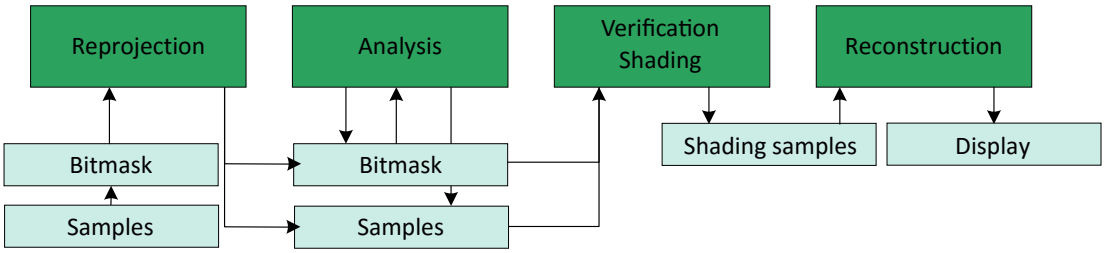


Figure 2: Main building blocks of our algorithm. Data are in light green, compute phases are in dark green.

3 OVERVIEW

In its most straight-forward implementation, stable ray tracing consists of four phases. *Reprojection* projects cached shading locations from the previous frame into screen space of the current frame, accounting for camera and object motion/deformation, to create a set of screen space sample locations. *Verification* constructs and traces primary visibility rays through the screen space sample locations to determine which of the reprojected shading positions are visible from the camera. Visible locations are then shaded, optionally caching intermediate results of the shading computation for later reuse. *Hole filling* generates screen space samples in regions where the density of visible reprojected points is low, and traces, shades, and caches hitpoint/shading information. Finally, *reconstruction* generates the final image for the current frame from the set of shaded samples.

The basic version of stable ray tracing improves temporal stability through the reuse of shading points across frames. However, there are a number of practical challenges to achieving interactive performance. In this section, we discuss these issues and associated tradeoffs, and briefly describe the choices we made in our system.

3.1 Sampling Rate and Uniformity

Sampling rate is the primary means of trading image quality for performance. Unlike conventional ray tracing, wherein screen sample locations are essentially independent of objects in the scene, in stable ray tracing screen space sampling density can be highly non-uniform due to the effects of camera and object movement on reprojected samples. Reprojection can lead to oversampling due to many points being reprojected to the same region of the screen, for example when an object moves away from the camera, or the camera zooms out. In such cases, maintaining performance requires that we ensure oversampling is kept to a minimum. Conversely, reprojection can also lead to undersampling due to disocclusions, or when sample density decreases due to a surface moving closer to the camera. In such cases, maintaining image quality requires that we ensure that enough samples are used. Highly non-uniform sampling can also lead to issues with resource contention (for example, multiple threads attempting to write to same cache location during reprojection) and load balancing. In addition, nonuniform sampling can produce artifacts when the sampling rate is very low compared to the reconstruction rate, as discussed in Section 7.

In order to ensure appropriate sampling rate and uniformity, our implementation adds an *analysis phase* prior to verification. The analysis phase efficiently estimates local sampling density and adds or removes samples to ensure the sampling rate falls within a specified range. As described in Section 4.2, the analysis phase makes use of a bitmask that encodes a quantized representation of the sampling pattern in each pixel, which allows us to estimate sampling density without having to read or recompute exact screen space locations for each sample.

3.2 Caching

Key to the efficiency and effectiveness of our implementation is the sample cache, which allows temporal re-use of shading locations and intermediate values. However, stable ray tracing's computational and memory overhead is proportional to the number of entries that are reprojected and potentially verified and shaded. As such, a cache eviction policy is needed that allows trading performance and memory use for temporal stability.

The simplest policy would be to evict points that are occluded or otherwise not used in the current frame. However, stability in the face of high-frequency visibility changes can be improved if occluded points remain in cache long enough to be re-used when they become visible again. As a result, there is a tradeoff between the space and reprojection cost of keeping occluded points in the cache and the temporal stability improvements to which such points may contribute in the future.

In addition to storing in the cache sufficient information to reconstruct world space position, we can also use the cache to avoid recomputation of expensive intermediate values required during shading (e.g., visibility or normals). Taken together, these values can cause each cache entry to be rather large. As such, minimizing overall size is important to performance, as is minimizing cache reads due to memory bandwidth constraints.

We use a two-phase cache eviction scheme that strives to strike a balance between overall performance and temporal stability. The first set of evictions occur in the reprojection phase (Section 4.1) and the second in the analysis phase (Section 4.2).

3.3 Ray Tracing

The basic stable ray tracing algorithm has two distinct ray tracing phases: verification and hole filling. The number of holes to be filled is typically small compared to the number of verification rays, and as a result the overhead associated with launching a separate hole-filling ray tracing pass can be non-trivial. As such, performance

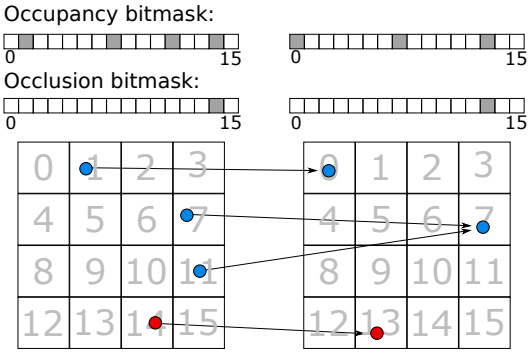


Figure 3: Reprojection phase for an M=4 set of subpixels from one frame, left, to the next, right, and evolution of the bitmasks. Both occluded and unoccluded samples are recorded in the occupancy bitmask. Occluded samples (red) also have a bit set in the occlusion bitmask. In the case of a collision between two samples (subpixel 7, right), the first unoccluded sample written to the subpixel is kept.

```
input : pixelDestination and subpixelDestination for a sample
        and associated data that isOccluded or not.
1  subpixel ← flatten (subpixelDestination);
2  bitOccupancy ← 1 ≪ subpixel;
3  bitOcclusion ← 1 ≪ (subpixel + M·M);
4  bitMask ← bitOccupancy ∨ (isOccluded? bitOcclusion: 0);
5  originalBitmask ← AtomicOr (pixelDestination, bitMask);
6  originalIsOccluded ← (bitOcclusion ∧ originalBitmask) ==
     bitOcclusion;
7  replace ← not isOccluded ∧ originalIsOccluded;
8  if not (isOccluded ∧ originalIsOccluded) then
9  |  AtomicAnd (pixelDestination, ¬bitOcclusion)
10 end
11 originalExists ← (bitOccupancy ∧ originalBitmask) ==
     bitOccupancy;
12 if replace ∨ not originalExists then
13 |  writeData( pixelDestination,data);
14 end
```

Algorithm 1: Pseudocode for sample reprojection storage.

could benefit if it were possible to combine the two ray tracing passes into one.

In our implementation, we fill verification-failure holes by using the occluding hitpoints discovered in the verification phase. This optimization improves performance over the naive implementation, at the cost of some sampling bias and an increase in sampling rate variance. However, the instability added is typically spatially incoherent and persists for a single frame, and as such is not usually objectionable.

4 METHOD

In this section, we discuss the details of our implementation, the design decisions we faced, and the choices we made. Our implementation is illustrated in Figure 2.

We store samples in two screen space data buffers, which serve as caches for the previous and current frame. At the beginning of each frame, samples are reprojected from the previous buffer to the current to account for object and camera motion. We analyze the outcome of the reprojection process and adaptively add or remove samples in the reprojection buffer in order to achieve a uniform sample distribution. The location samples are then verified, and finally shaded. The resulting color information is stored in a shading buffer, which is used by the reconstruction phase to resolve color.

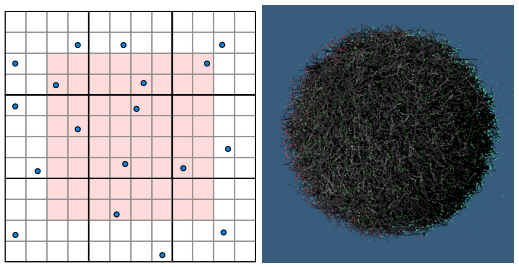
4.1 Reprojection

Stable ray tracing requires that cached samples are updated to reflect scene dynamics such as camera motion and object motion and deformation. The data to be stored per sample in the reprojection buffers should thus be chosen according to the scene dynamics that one would like to support. We store a 3D position in object space coordinates and a transform ID to support affine transformations. More data would likely be required to support arbitrary object deformation. The ID we store is used to access an object-to-world transformation matrix for the current frame. This matrix is in turn used to transform the sample position to world space. We then project the world space position onto the screen using the current camera transformation, and we clip away samples that fall outside the screen area.

During reprojection, we take steps to ensure that not too many samples reproject to the same screen location in order to reduce resource contention, improve load balancing, and manage size of the cache. We also strive to preferentially keep samples that are visible over those that are occluded.

To do so, we divide each pixel in the reprojection buffer into $M \times M$ subpixels, as illustrated in Figure 3. We maintain a corresponding occupancy bitmask representing the occupancy state of each subpixel, which is cleared at the start of each frame. The occupancy bitmasks are also used during the analysis phase to determine approximate sample location and local sample density. We similarly maintain with each pixel an $M \times M$ bitmask that indicates if the sample in each subpixel is occluded; values in this occlusion bitmask are written during the verification phase. Storing these bitmasks separately from the cache values themselves allows us to reduce bandwidth required by the reprojection phase.

When a source sample reprojects into a given destination subpixel, we check the destination subpixel's corresponding occupancy bit in the bitmask. If the destination subpixel occupancy bit is zero, the sample is written to the destination location, the destination occupancy bit is set to one, and the destination subpixel occlusion bit is copied from the source bitmask. If the destination subpixel occupancy bit is one, we examine the destination subpixel occlusion bit. If the destination subpixel occlusion bit is one and the source occlusion bit is zero, the source sample is written to the destination, and the destination occlusion bit set to zero. Otherwise the source sample is not written to the destination buffer, effectively evicting



analysis extent update pattern

Figure 4: Left: the spatial extent (light red) of our local density analysis. Right: Update pattern that arises from our analysis scheme. Example: hairball sequence with a right-rotating camera and $d_{\text{target}} = d_{\text{tolerance}} = 1$. Green pixels indicate areas where new samples are added, while red samples indicate where samples are evicted.

it from the cache. A pseudocode outline of our eviction scheme is in Algorithm 1.

Data races due to competing threads working on the same sample can be avoided by atomically updating the per-sample data, potentially causing a large performance impact. We note instead that as we only perform atomic updates of the bitmasks a data race can only occur when a first occluded sample lands on a sample and second unoccluded one tries to overwrite it. In this rare case, we would simply store the occluded sample over the unoccluded, leading to reduced temporal stability. In practice we found these events to be rare and to have small impact on the final image quality.

Our sample rejection policy ensures that we cache at most $M \times M$ samples in any pixel, enforcing an upper bound on storage and subsequent processing costs, while maintaining a good screen-space distribution of samples, unlike, for example, simply keeping the first $M \times M$ samples that reproject into a given pixel would. The mechanism also ensures that unoccluded samples are preferentially cached over occluded samples.

4.2 Sample Analysis

In regions that are oversampled, analysis chooses which samples to remove, and adds new samples in undersampled regions to meet the desired sampling rate.

To help ensure a good spatial distribution of samples, we divide each pixel in a number of strata (in our implementation, 4). For each stratum, we count the number of samples. To remove samples, we choose from the substratum with the most number samples, selecting randomly in the case of a tie. Similarly, we progressively add samples to the substratum with the fewest samples. This process allows us to stratify the samples across the pixel. Within a substratum, new samples are placed in the center, with a small random offset in order to avoid correlation in the screen space location of the samples.

To minimize the overall performance impact of analysis, we use the occupancy and occlusion bitmasks to determine whether samples should be added or removed. To determine how many to add or remove, we analyze the local sample density $d = N/A$, where N is the number of unoccluded samples in an area of $A = 2 \times 2$ pixels around the current pixel. The user can then specify two parameters, d_{target} and $d_{\text{tolerance}}$. The algorithm will not add or remove samples if the density is within $[d_{\text{target}} - d_{\text{tolerance}}, d_{\text{target}} + d_{\text{tolerance}}]$. Otherwise, we add or remove enough unoccluded samples ΔN to bring the density within limits:

$$\Delta N = \begin{cases} \operatorname{sgn}(d_{\text{target}} - d) \left\lceil \left| d_{\text{target}} - d \right| \right\rceil & \text{if } \left| d_{\text{target}} - d \right| \geq d_{\text{tolerance}} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where the sign of ΔN tells us whether we need to add or remove samples. Figure 4 illustrates a typical pattern of sample addition and removal for a dynamic scene.

It is necessary to modify the cache when we add a new sample, since in the next phase we need to distinguish between new and cached samples. To remove a sample, we simply set the corresponding occupancy bit to zero. For a new sample, we write $(\text{NaN}, \mathbf{p}^x, \mathbf{p}^y)$ instead of its object space position. The NaN marks the sample as new. Since we have to store the new sample in memory, we also store the chosen screen space coordinates for the sample $(\mathbf{p}^x, \mathbf{p}^y)$.

4.3 Verification and Shading

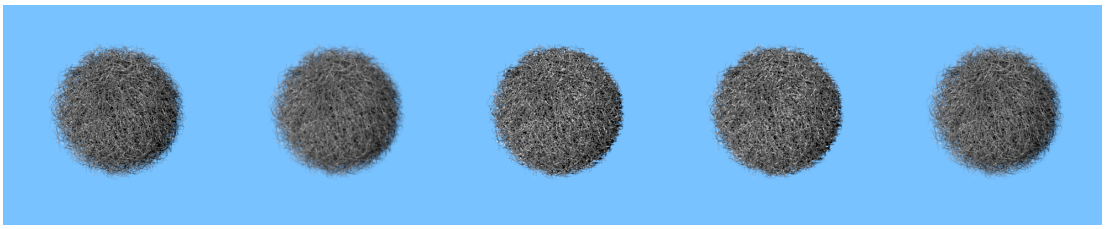
The verification phase processes the location samples to generate shading samples for the reconstruction phase. Our algorithm works on top of any ray tracing framework that provides programmable camera and closest hit stages. We define a standard ray as a tuple $\mathbf{r} = (\mathbf{o}, \vec{d}, t_{\text{min}}, t_{\text{max}})$, where the quantities represent origin, direction, and minimum and maximum intersection distances, respectively.

In this step, we distinguish between cached samples and newly generated samples with screen space coordinates $(\text{NaN}, \mathbf{p}^x, \mathbf{p}^y)$ in the cache. We trace these new samples with a closest hit ray, using the stored screen space position to generate a corresponding world space direction $\vec{d}$ according to our camera model. Given the camera position $\mathbf{c}$, our ray becomes $\mathbf{r} = (\mathbf{c}, \vec{d}, \epsilon, +\infty)$. Once the ray tracing operation terminates, we store the hitpoint object space position and transform ID in the reprojection cache, and the corresponding shade in the shading cache.

For existing samples with cached position $\mathbf{x}_{\text{object}}$, we first compute its corresponding world space position $\mathbf{x}_{\text{world}}$. Then, we cast a closest hit ray $\mathbf{r}_{\text{cached}} = (\mathbf{c}, (\mathbf{x}_{\text{world}} - \mathbf{c})/\|\mathbf{x}_{\text{world}} - \mathbf{c}\|, \epsilon, \|\mathbf{x}_{\text{world}} - \mathbf{c}\| + \epsilon)$. When we hit the closest surface, we verify that the sample is still visible in the current frame. If the sample is still visible, the intersected t should match the cached $t = \|\mathbf{x}_{\text{world}} - \mathbf{c}\|$.

Occluded samples can cause numerical instability in the shading distribution, in particular around geometric edges. In our implementation, we normally mark such samples as occluded and keep them in the cache. However, if an occluded sample is the last one remaining in a pixel, we replace its hitpoint with the one from the occluding surface. This allows us to maintain a minimum sample density without requiring a new ray to be traced, as discussed in Section 3.3.

Once a sample is verified, or if it is new, we shade it according to our rendering algorithm, and store the results in the shading buffer, alongside its subpixel position.



Supersampling, 4 spp	Supersampling, 4 spp + temporal antialiasing	Stable ray tracing, 2 spp	Stable ray tracing, 2 spp + temporal integration	Supersampling, 32 spp
sharpness: 0.8142	sharpness: 0.6610	sharpness: 0.8056	sharpness: 0.7783	sharpness: 0.8054

Figure 5: Comparison of frames rendered for the hairball video. For each technique, we report the number of samples per pixel (spp) and the CPBD-based image sharpness. Stable ray tracing strikes a compromise between sharpness and temporal stability at the price of added spatial aliasing.

Technique	Reprojection	Analysis	Verification / Shading	Reconstruction	Total
Stable ray tracing, $d_{\text{target}} = 1$	1.05 ms	0.28 ms	18.91 ms	0.71 ms	**20.94 ms**
Stable ray tracing, $d_{\text{target}} = 2$	1.23 ms	0.38 ms	28.88 ms	0.82 ms	**31.31 ms**
Stable ray tracing, $d_{\text{target}} = 4$	1.73 ms	0.62 ms	47.48 ms	0.90 ms	**50.73 ms**
Supersampling, 1 spp	-	-	13.35 ms	0.21 ms	**13.56 ms**
Supersampling, 2 spp	-	-	20.94 ms	0.38 ms	**21.32 ms**
Supersampling, 3 spp	-	-	28.36 ms	0.54 ms	**28.90 ms**
Supersampling, 4 spp	-	-	35.86 ms	0.71 ms	**36.57 ms**
Supersampling, 5 spp	-	-	43.40 ms	0.88 ms	**44.28 ms**
Supersampling, 6 spp	-	-	50.91 ms	1.04 ms	**51.95 ms**

Table 1: Average time spent per frame in the hairball video for each phase of the different techniques. All results use GPU timers. The additional price for stable ray tracing is a slowdown of the overall rendering time between 1.4x and 1.5x. Temporal integration is performed on the resulting image, at an additional cost of 0.67 ms.

4.4 Reconstruction

Each color sample stored in the previous step carries an RGB color and subpixel position. We then filter our resulting color using a 3×3 truncated spatial Gaussian filter. Our algorithm does not guarantee uniform sampling rate, since it trades off a uniform rate for temporal stability. A nonuniform sampling density can lead to challenges in reconstruction, such as pixels with no samples. At low sampling densities, the use of this simple reconstruction filter can lead to blurring and apparent thickening of edges. We discuss the artifacts resulting from trading spatial uniformity for temporal coherence in Section 7.

After reconstruction, an additional post processing step may be performed. In Section 6, we discuss how our method fares with a temporal reconstruction scheme on top, namely temporal integration. When performing this additional step, we calculate and store motion vectors in the shading cache, picking the one with maximum length during reconstruction.

5 IMPLEMENTATION DETAILS

Our reprojection and analysis phases are implemented as OpenGL compute shaders. The reprojection shader transfers data between two identically deep screen sized buffers. The verification and shading step is implemented on the GPU in the camera program using the NVIDIA OptiX ray tracing engine [2010]. The programmable ray tracing pipeline of OptiX allows us to insert our cache management. The reconstruction and post processing were implemented as full screen passes in OpenGL shaders.

We compress our samples as 16-bytes elements of which 12 are reserved for 3 floating point elements defining position in object space. Due to OpenGL-OptiX interoperability limitations, we were not able to write the occlusion bit in the bitmask in the verification and shading phase directly. So we use one of the remaining 32 bits to store occlusion for the sample. Note that this does not change performance, since we have to fetch the sample anyways in the reprojection phase. The remaining 31 bits are reserved for a transform ID to allow affine transformations. The existence and occlusion statuses of the samples are stored in the bitmasks, for which we use $M = 4$. We use the two halves of a 32 bits unsigned integer to store both 16 bits bitmasks. The shading samples are stored as 8 bytes elements: 3 bytes for the tone-mapped color, 4 bytes for a motion vector (16 bits per component) and 1 byte for the subpixel position and flags (3+3 bits for position in a 8x8 grid, plus 1 bit for an existence flag).

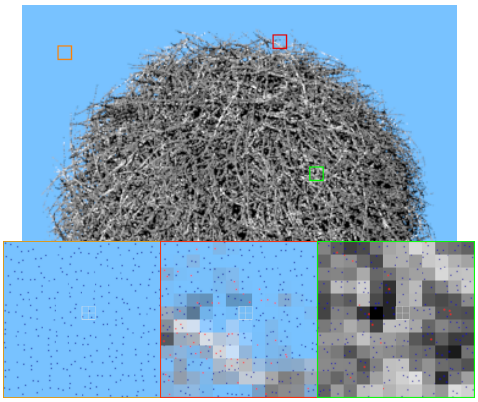
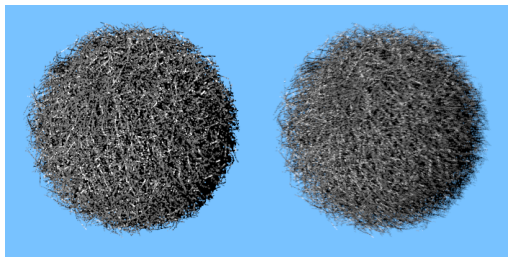


Figure 6: Three examples of sample distributions generated by our algorithm in the areas marked by the colored squares. Blue circles represent visible samples, red circles represent occluded samples.



Stable ray tracing, 1 spp
sharpness: 0.8182

[Martin et al. 2002]
sharpness: 0.6957

Figure 7: Quality comparison with Martin et al. [Martin et al. 2002] for frame 526 of the martin_comparison.mp4 video. Their technique produces a blurrier result and is also more temporally unstable.

6 RESULTS

Given the dynamic nature of our algorithm, we provide some of our results in a video (hairball.mp4) of a static hairball [McGuire 2011] captured with a moving camera. The hairball has a standard glossy material applied, and is illuminated by a single point light to which a shadow ray is traced per shading evaluation. The frames of the video were captured individually and then assembled to create a video of 60 frames per second. All our results were generated using an NVIDIA GeForce GTX 1080 graphics card. We report rendering times for a 1080x1080 image frame.

The hairball video compares stable ray tracing with supersampling of similar performance. In addition, to measure the impact of a recent temporal noise reduction scheme, we apply temporal integration with color clamping in the variant proposed by Patney et al. [2016]. For stable ray tracing, samples are not jittered and we choose an integration factor of $\alpha = 0.25$. For supersampling, we use $\alpha = 0.1$ and do full temporal antialiasing by including sample jittering in the temporal integration. The larger α used for stable ray tracing incurs a smaller amount of blur, which we can get away with because our input values are more stable. If we use $\alpha > 0.1$ with supersampling, the temporal antialiasing cannot hide the underlying temporal instability. A single frame of the hairball video is provided in Figure 5. Here, we compare image sharpness using a CPBD-based sharpness metric [Narvekar and Karam 2011]. The sharpness score measures the percentage of edges at which blur (probably) cannot be detected. The video shows a reference rendering, rendered as 32 samples per pixels.

Comparing supersampling and stable ray tracing, we first observe that while stable ray tracing does not completely remove temporal artifacts (in particular around the strands of the hairball), the final result perceptually improves in temporal stability. This is especially true at the beginning of the video, where the camera is only rotating. Sharpness of stable ray tracing and supersampling is similar to that of the reference, with supersampling being slightly greater than reference. Once we apply temporal integration to both results, the situation reverses. Supersampling with temporal integration is more temporally stable (although some underlying noise is still present), but it is also significantly more blurry. Temporal integration applied to stable ray tracing reduces some of the higher frequency noise, but it also better preserves sharpness while retaining temporal stability.

Since our algorithm trades temporal stability for an irregular spatial sampling pattern, we want to validate the aliasing artifacts that are generated by the algorithm. An example of the kind of distribution of samples we achieve with our algorithm is shown in Figure 6, for three different areas of a single frame of the hairball video. We compare the quality for different target densities of our algorithm in Figure 8, for three different scenes (hairball, plane with text and ogre). The images were taken after 25 frames of an animated video, to allow stable ray tracing to set into a nonstandard sampling pattern. We provide closeups to better show the artifacts generated at a pixel level. For the lowest sample count (1 spp averages), we can see that stable ray tracing introduces artifacts. In the hairball frames, we can see that this manifests as thickened edges. In the plane with text frame, the artifacts manifest as broken edges and letters. In the ogre scene, they manifest as weirdly shaped specular highlights. For averages of 2 spp, the differences reduce and it almost disappears with averages of 4 spp.

We compare the performance of stable ray tracing against supersampling in Table 1. All results were obtained using OpenGL GPU timers, averaging the milliseconds spent in each phase over the whole sequence in the hairball video. From the totals in the table, we can see that stable ray tracing generally performs 1.4 to 1.5 times slower than the equivalent supersampling. This is similar to the performance cost of a factor of around 1.35 reported by Martin et al. [2002]. The overhead of reprojection and analysis phases is between 1 and 3 milliseconds. We note that the reconstruction phase for stable ray tracing has a higher impact than the one in supersampling, given that we need to adapt it to the irregular number of samples we have per pixel.

We made a comparison with Martin et al. [2002], tweaking the algorithm to fit modern GPU pipelines. For each sample, we generate

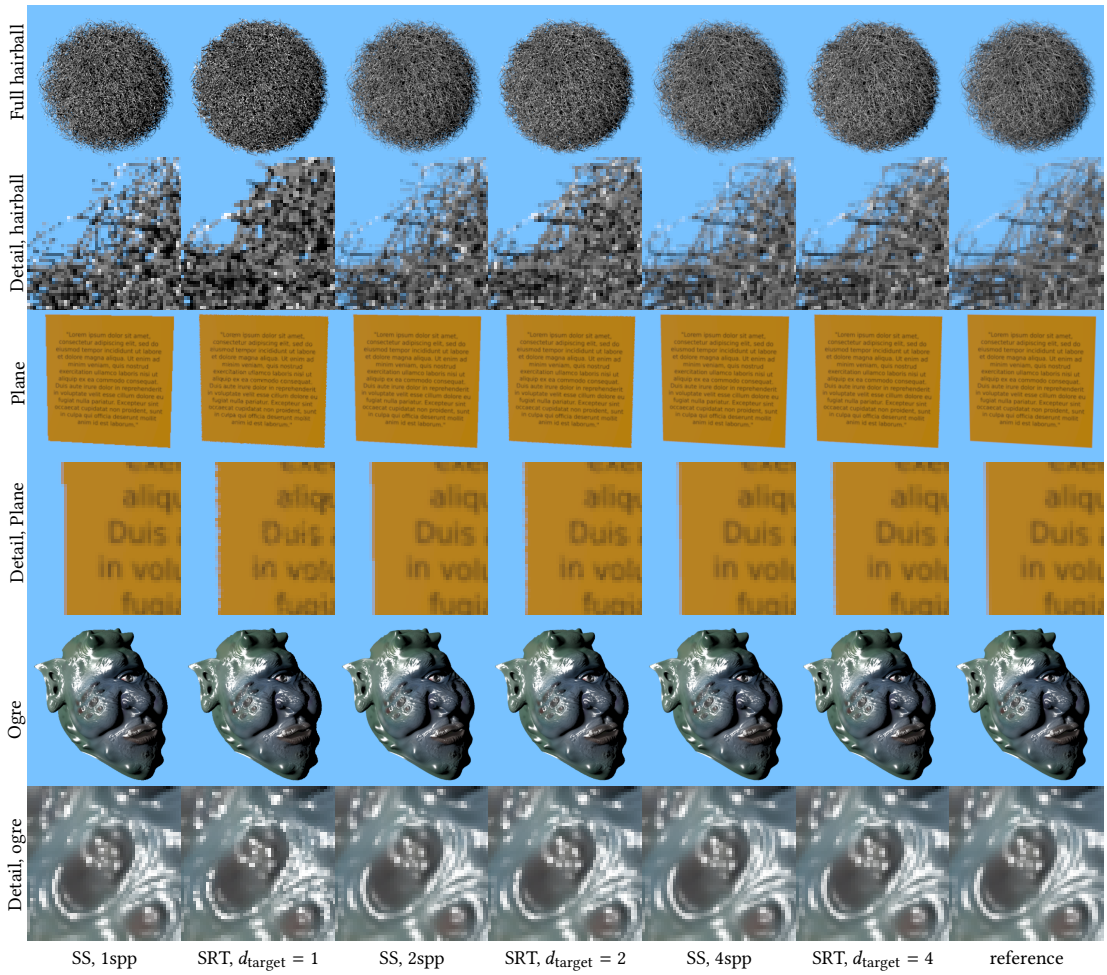


Figure 8: Quality comparisons between standard supersampling (SS) and stable ray tracing (SRT), for different number of samples. We use the same Gaussian reconstruction filter for all images. For low sample counts, stable ray tracing gives a result that is more temporally stable, at the price of introducing spatial aliasing artifacts.

a single vertex, rendered as a 1x1 pixel splat in the final destination pixel. This allows us to use the depth buffer to find the closest sample in the case of multiple samples landing in one pixel. If a sample does not exist, we simply generate a vertex outside of the view frustum. Then, a ray tracing step generates a sample in the middle of the pixel if it does not find one, and traces the ray. Relatively, our implementation is a bit faster than the original method, being only 1.2 times slower than the equivalent supersampling. The results are in a video (martin_comparison.mp4) and in Figure 7, where we provide a comparison with our method for similar sample counts. On the left-hand side of the video, we compare the two techniques for a panning view of a bump mapped plane. In this case, the quality of the two techniques is similar, except for the blurring due to the temporal filter employed by Martin et al. [2002]. If we consider the hairball (right-hand side of the video), our method is significantly more temporally stable. In addition, since we do not use an averaging temporal filter, our method produces sharper images (see Figure 7).

6.1 Application: Progressive Path Tracing

Our screen space sample data structure serves a double purpose: nearby samples in the data structure are close in world space, and

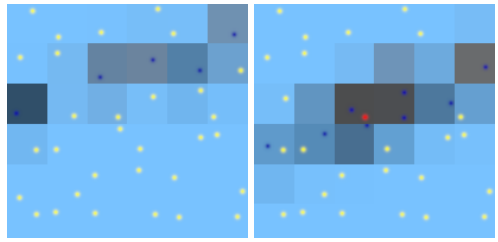
Supersampling, 2 spp	Supersampling, 2 spp + temporal antialiasing	Stable ray tracing, 1 spp	Stable ray tracing, 1 spp + temporal integration	Supersampling, 32 spp
sharpness: 0.7924	sharpness: 0.5348	sharpness: 0.7085	sharpness: 0.6060	sharpness: 0.6771

Figure 9: Including indirect illumination, the different techniques are here applied to a frame in the Sponza video.

the majority of samples are consistent in world space across frames. These properties make stable ray tracing suitable for accumulating view-independent but time-dependent information, such as diffuse indirect illumination.

As a proof of concept, we apply our technique on top of standard unidirectional path tracing to cache diffuse indirect illumination in a dynamic scene. For performance reasons, our path tracing has a fixed maximum trace depth. For each frame, we choose a random direction, trace a new path in that direction, and accumulate the final result. Directions are sampled using a cosine-weighted hemispherical distribution. For a completely static scene, we could give equal importance to all frames. Since we want to be able to react to dynamic content in the scene, we use a simple exponential moving average [Nehab et al. 2007; Scherzer et al. 2007] with integration factor 0.1. Our focus is here to illustrate the virtues of stable ray tracing in accumulation. More complicated sampling schemes are possible, such as accumulating indirect illumination to allow convergence when camera and scene are static, or from literature [Herzog et al. 2010; Yang et al. 2009].

Like the hairball video, we provide a similar video comparison for our global illumination method. In this video (sponza.mp4), we compare our progressive path tracing to a similar performance supersampling with 2 samples per pixel. As in the previous section, we provide comparisons with and without the temporal integration schemes. In this comparison, we observe that stable ray tracing improves temporal stability for a scene with a dynamic moving light. Some noise is still present, mostly due to fireflies generated by the new shading directions chosen for each sample. Since we use a screen space data structure, results must be re-generated upon disocclusion. This is why the flagpoles in the video leave trails of higher variance content. Figure 9 compares a cutout of a still frame of the Sponza video. One should note that the blur incurred by the temporal post-processing filters is good at hiding the stochastic noise of the path tracing. However, as is clear from visual comparison and the CPBD-based image sharpness measurements, the blurring of temporal antialiasing on top of supersampling is too much. We also note how the reference rendering in this case also has a lower sharpness score than results without temporal post-processing filters. This is mainly due to the noise in these two images being considered as sharpness.



initial distribution thickened edge

Figure 10: Edge thickening. Blue samples belong to one of the hairball strands, yellow samples belong to the background, and red samples are occluded. In the right image, the camera has moved upwards, so that the apparent motion in screen space of the edge is downwards. The low local density in the area above the strand causes the thickening.

7 DISCUSSION

Stable ray tracing improves temporal stability while retaining sharpness (hairball video and Figure 5). Our algorithm offers an intermediate solution between supersampling, which is sharp but temporally unstable, and temporal antialiasing, which is too blurry. The reason for this excessive blurriness is the high temporal instability in the input from supersampling. Since we do not have this temporal instability, we can apply a more relaxed temporal filtering (larger α) and thus strike a compromise between stability and sharpness. On the other hand, we cannot use jittering and therefore pay the price of spatial aliasing artifacts. These artifacts are particularly evident at lower sampling rates, resulting in broken or thickened edges and changed highlight patterns (Figure 8).

Spatial aliasing artifacts arise from the fact that we do not estimate the screen space coverage of each sample, but rather give them the same weight in the reconstruction phase. As we illustrate in Figure 10, this causes edge thickening. The distribution of samples changes a bit, but not enough to change the density. New samples are therefore added. The small gap introduced by the change in distribution is filled as possible by the reconstruction algorithm, causing the edge to thicken. A lower $d_{\text{tolerance}}$ could mitigate this

problem by fixing the distribution more quickly wherever necessary. However, lowering this parameter would cause samples to get recycled more often, leading as well to temporal instability. This screen space coverage problem is partly to blame for the residual temporal instability of stable ray tracing, since each sample would have a different estimated coverage every frame.

As previously noted, the overhead of our technique is similar to that of Martin et al. [2002]. In our video comparison, we see how we reduce temporal artifacts, by allowing an irregular number of samples per pixel in our technique. This allow us to remove the originally proposed scene-based temporal filter, increasing the sharpness of the final image in the process. Although the overhead added by the reprojection and analysis phases are relatively low, there is an additional verification overhead when comparing on an iso-sample-rate basis. This penalty is due to load balancing issues resulting from the nonuniform screen-space sampling patterns, and subsequent varying amount of per-pixel work, generated by reprojection. We expect that the ray tracing overhead can be reduced by performing a load balancing step prior to tracing rays.

Our Sponza video exemplifies the potential of stable ray tracing as a technique for caching indirect light. In this example, due to the nature of our accumulation scheme, the fireflies generated by the path tracing procedure cause an additional level of temporal instability. However, our algorithm still retains its qualities, retaining a higher temporal stability (at least when temporal filtering is not used to hide it) and better image sharpness (Figure 9).

8 CONCLUSION

We presented a new practical technique for stable shading in interactive ray tracing. Our technique is based on sample reprojection and introduces low cost sample analysis for generating and evicting samples in the reprojection cache. The stable ray tracing that we propose is useful for striking a balance between temporal stability and image sharpness in interactive ray tracing applications. This comes at the cost of spatial aliasing and around a factor 1.5 hit to the performance. If the rendering budget allows a target sample density of just 4 samples per pixel, our technique can eliminate most spatial aliasing artifacts and provide a visually pleasing (sharp, antialiased) and fairly temporally stable result. Since we have stable shading in a ray tracing context, we can use our shading cache to add global illumination effects such as progressively path traced indirect illumination. In general, our algorithm eases the use of progressive techniques when a scene is dynamic.

REFERENCES

Stephen J. Adelson and Larry F. Hodges. 1995. Generating exact ray-traced animation frames by reprojection. *IEEE Computer Graphics and Applications* 15, 3 (1995), 43–52.

Sig Badt, Jr. 1988. Two algorithms for taking advantage of temporal coherence in ray tracing. *The Visual Computer* 4, 3 (1988), 123–132.

Kavita Bala, Bruce Walter, and Donald P. Greenberg. 2003. Combining edges and points for interactive high-quality rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)* 22, 3 (July 2003), 631–640.

John Chapman, Thomas W. Calvert, and John Dill. 1991. Spatio-temporal coherence in ray tracing. In *Proceedings of Graphics Interface (GI '91)*. 101–108.

Shenchang Eric Chen and Lance Williams. 1993. View interpolation for image synthesis. In *Proceedings of SIGGRAPH 93*. ACM, 279–288.

Robert L. Cook, Loren Carpenter, and Edwin Catmull. 1987. The Reyes image rendering architecture. *Computer Graphics (Proceedings of SIGGRAPH 87)* 21, 4 (July 1987), 95–102.

Henri Gouraud. 1971. Continuous shading of curved surfaces. *IEEE Trans. Comput.* 20, 6 (June 1971), 623–629.

Eduard Gröller and Werner Purgathofer. 1991. Using temporal and spatial coherence for accelerating the calculation of animation sequences. In *Proceedings of Eurographics (EG '91)*, Vol. 91. 103–113.

Vlastimil Havran, Cyrille Damez, Karol Myszkowski, and Hans-Peter Seidel. 2003. An efficient spatio-temporal architecture for animation rendering. In *Rendering Techniques 2003 (Proceedings of EGSR 2003)*, Per H. Christensen and Daniel Cohen-Or (Eds.). Eurographics Association, 106–117.

Robert Herzog, Elmar Eisemann, Karol Myszkowski, and Hans-Peter Seidel. 2010. Spatio-temporal upsampling on the GPU. In *Proceedings of Interactive 3D Graphics and Games (I3D '10)*. ACM, 91–98.

Jose A. Iglesias-Guitian, Bochang Moon, Charalampos Koniaris, Eric Smolikowski, and Kenny Mitchell. 2016. Pixel history linear models for real-time temporal filtering. *Computer Graphics Forum (Proceedings of Pacific Graphics 2016)* 35, 7 (October 2016), 363–372.

Jorge Jimenez, Jose I. Echevarria, Tiago Sousa, and Diego Gutierrez. 2012. SMAA: enhanced subpixel morphological antialiasing. *Computer Graphics Forum (Proceedings of Eurographics 2012)* 31, 2pt1 (May 2012), 355–364.

Brian Karis. 2014. High-quality temporal supersampling. In *Advances in Real-Time Rendering in Games, Part I.* Number 10 in ACM SIGGRAPH 2014 Courses. http://advances.realtimerendering.com/s2014/

William R. Mark, Leonard McMillan, and Gary Bishop. 1997. Post-rendering 3D warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics (I3D '97)*. ACM, 7–16.

William Martin, Peter Shirley, Steven Parker, William Thompson, and Erik Reinhard. 2002. Temporally coherent interactive ray tracing. *Journal of Graphics Tools* 7, 2 (2002), 41–48.

Morgan McGuire. 2011. Computer Graphics Archive. (August 2011). http://graphics.cs.williams.edu/data

Koichi Murakami and Katsuhiko Hirota. 1992. Incremental ray tracing. In *Photorealism in Computer Graphics (Proceedings of EGWR 1990)*, K. Bouatouch and C. Bouville (Eds.). Springer, 17–32.

N. D. Narvekar and L. J. Karam. 2011. A no-reference image blur metric based on the cumulative probability of blur detection (CPBD). *IEEE Transactions on Image Processing* 20, 9 (September 2011), 2678–2683.

Diego Nehab, Pedro V. Sander, Jason Lawrence, Natalya Tatarchuk, and John R. Isidoro. 2007. Accelerating real-time shading with reverse reprojection caching. In *Proceedings of Graphics Hardware (GH 2007)*. 25–36.

Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. 2010. OptiX: a general purpose ray tracing engine. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)* 29, 4 (July 2010), 66:1–66:13.

Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. 2016. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)* 35, 6 (November 2016), 179:1–179:12.

Daniel Scherzer, Stefan Jeschke, and Michael Wimmer. 2007. Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In *Rendering Techniques 2007 (Proceedings of EGSR 2007)*. Eurographics Association, 45–50.

Pitchaya Sitthi-amorn, Jason Lawrence, Lei Yang, Pedro V Sander, and Diego Nehab. 2008a. An improved shading cache for modern GPUs. In *Proceedings of Graphics Hardware (GH 2008)*. Eurographics Association, 95–101.

Pitchaya Sitthi-amorn, Jason Lawrence, Lei Yang, Pedro V Sander, Diego Nehab, and Jiahe Xi. 2008b. Automated reprojection-based pixel shader optimization. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2008)* 27, 5 (December 2008), 127:1–127:11.

Takehiro Tawara, Karol Myszkowski, Kirill Dmitriev, Vlastimil Havran, Cyrille Damez, and Hans-Peter Seidel. 2004. Exploiting temporal coherence in global illumination. In *Proceedings of Spring Conference on Computer Graphics (SCCG 2004)*. ACM, 23–33.

Edgar Velázquez-Armendáriz, Eugene Lee, Kavita Bala, and Bruce Walter. 2006. Implementing the render cache and the edge-and-point image on graphics hardware. In *Proceedings of Graphics Interface 2006 (GI '06)*. Canadian Information Processing Society, 211–217.

Bruce Walter, George Drettakis, and Donald P. Greenberg. 2002. Enhancing and optimizing the render cache. In *Proceedings of the Eurographics Workshop on Rendering (EGWR 2002)*. ACM Press, 37–42.

Bruce Walter, George Drettakis, and Steven Parker. 1999. Interactive rendering using the render cache. In *Rendering techniques '99 (Proceedings of EGWR 1999)*. Springer, 19–30.

Lei Yang, Diego Nehab, Pedro V. Sander, Pitchaya Sitthi-amorn, Jason Lawrence, and Hugues Hoppe. 2009. Amortized supersampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2009)* 28, 5 (December 2009), 135:1–135:12.

Peng Zhou and Yanyun Chen. 2015. Variance reduction using interframe coherence for animated scenes. *Computational Visual Media* 1, 4 (December 2015), 343–349.

Tenghui Zhu, Rui Wang, and David Luebke. 2005. A GPU accelerated render cache. In *Proceedings of Pacific Graphics 2005 (short paper)*.

Appendix V

Virtual Reality inspection and painting with measured BRDFs

Virtual Reality inspection and painting with measured BRDFs

Alessandro Dal Corso
Technical University of Denmark

Jonathan Dyssel Stets
Technical University of Denmark

Andrea Luongo
Technical University of Denmark

Jannik Boll Nielsen
Technical University of Denmark

Jeppe Revall Frisvad
Technical University of Denmark

Henrik Aanæs
Technical University of Denmark

Figure 1: Pictures illustrating our VR demo application, with an in-game screenshot (left) and a picture of the setup (right). Paintbucket, table and lightbuld models ©TurboSquid.com, environment maps ©HDRMaps.com and ©Joost Vanhoutte.

CCS CONCEPTS

• **Computing methodologies** → **Virtual reality**;

KEYWORDS

Virtual Reality, material appearance

ACM Reference Format:
Alessandro Dal Corso, Jonathan Dyssel Stets, Andrea Luongo, Jannik Boll Nielsen, Jeppe Revall Frisvad, and Henrik Aanæs. 2017. Virtual Reality inspection and painting with measured BRDFs. In *Proceedings of SA '17 VR Showcase, Bangkok, Thailand, November 27-30, 2017,* 2 pages.
https://doi.org/10.1145/3139468.3139472

1 INTRODUCTION

This is a virtual reality (VR) painting application that enables the user to paint on 3D models with real measured materials, much like in the physical world. A scanned physical object can be imported into the VR application, and the user can paint on the surface of the object with a virtual hand-controlled paint brush. The user is presented with several paint buckets, each containing a material known from the physical world. These materials are measured bidirectional reflectance distribution functions (BRDFs) of real physical materials. The materials and objects present in VR are thus represented as they would be in the physical world, and the user can control both the environment lighting and a single light source. The application enables analog artists to apply their skills directly on a digital model and it enables engineers to directly inspect BRDFs in a fast and intuitive way. Figure 1 is a screenshot from the application showing models that have been painted with BRDFs.

2 MOTIVATION

The realization of our VR demo was mainly driven by two objectives. The first was to have an intuitive and practical way of inspecting and visualizing measured BRDFs. Our application enables the user to apply BRDFs to 3D objects in a simple way that mimics the action of painting in real life. The user can also interact with the environment and the objects through an intuitive interface. Some of the actions the user can perform consist of modifying the light source intensity and position, the environment map, and the position and orientation of the objects. All these features allow us to quickly visualize and inspect a chosen BRDF under different lighting conditions without having to interact with a complicated interface.

Our second objective was to have an application useful in industrial and artistic design [Wald et al. 2006]. We provide a new way for artists, both digital and analog, to transfer their skills to the 3D domain by painting materials directly on 3D models that they might have created or scanned from real objects. By providing a real-time rendering environment, we enable artists to immediately see the final results of their creations under different viewing perspectives and lighting conditions. With this VR application, we aim at moving

SA '17 VR Showcase, November 27-30, 2017, Bangkok, Thailand

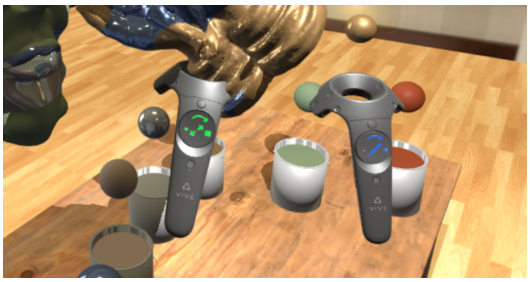


Figure 2: Interaction (left) and paintbrush (right) controller. Note the icons allowing to change the model and the brush size, respectively. See Figure 1 for copyright notice.

a first step towards the creation of a tool that will allow artists to apply materials and colors in a way that is similar to what they would do in the real world. Thus, we support the more traditional artistic workflow rather than inventing new artistic workflows for editing BRDFs [Colbert et al. 2006].

A user entering the virtual world will see a simple scene: a table, calibrated to match the position and height of a table present in the real world, a set of paint buckets containing different materials based on measured BRDFs, several 3D models, and a light-bulb. The user interacts with the scene through two handheld trackable controllers, one used as a painting tool and the other one as a grabbing and interaction tool. He/she can grab, move, rotate, and scale the 3D models with one hand and paint and select a different material with the other. The user can also move the position of the light-bulb and its intensity to get an immediate feedback on the appearance of their work. Furthermore, we use haptic feedback to enhance the interaction between user and objects in the scene. Our application thus creates a bridge between the digital and the physical domain, with an interface known from the physical world that the user is already familiar with. All these features help users immerse themselves and unfold their creativity in an environment similar to what they would see in a real artist's studio, and we have aimed to make this environment as interactive as possible.

3 DETAILS

Our demo is an HTC Vive setup (https://www.vive.com/), using the two provided controllers as interaction devices. One of the controllers is used for interaction, while the other acts as a paintbrush. The interaction is activated using the trigger button on the controller, and is used for grabbing and moving objects, including the light bulb above the table and the paint buckets. We use the small spheres on the left side of the table to change environment map. Finally, we have an undo button on the right to undo/redo the last action performed. Dipping the paintbrush into a bucket changes the BRDF that it will paint with. Based on the controller touchpads (see Figure 2), we also added interactions for changing object size (while grabbed), paintbrush size (on the paintbrush controller), and light intensity (while the light is grabbed).

We use measured BRDFs both from the MERL database [Matusik et al. 2003] and from our own laboratory. The scene has two light sources: an environment map and a movable point light in the form of a light bulb. The environment map contributes with background, reflections, and an ambient term. The ambient term is computed through standard spherical harmonics multiplied by the bihemispherical reflectance ρ of each measured BRDF, calculated in a preprocessing step using Monte Carlo integration. We multiply the environment map reflected color by ρ and by the factor $\min(1, \frac{f_{\max}}{C\rho_{\text{avg}}} - 1)$, where $f_{\max}$ is the peak value in the measured material, ρ_{avg} is the average of the three channels of ρ, and C is a user-defined constant. In case of a material with a strong reflection peak (such as a metallic paint), $\rho_{\text{avg}} \ll f_{\max}$ and the factor will be equal to one. In a case of a more diffuse material, $\rho_{\text{avg}} \approx f_{\max}$ and the reflection term will not be included. To paint the material, we intersect a sphere with the vertices of the model. For materials without a UV map, we write a material label on a per-vertex data structure. If a UV map is present, we first generate a secondary texture that maps vertex coordinates to UVs and then write the labels into a texture using the generated mapping.

4 USER FEEDBACK AND CONCLUSION

We invited an analog artist, a design engineer, and two 3D artists to test our application. They all found the interface intuitive to work with. Most noticeably, the analog artist used the full system without any previous VR experience after a one minute verbal instruction. This supports the objective of our application to enable transfer of artistic skills from the analog to the digital domain. Users noted the bulkiness of the controller compared to real-life painting tools such as a brush or a pencil. We accept this limitation of the system, hoping for smaller, lighter, or more customizable solutions in the future, like the stylus presented by Jackson and Keefe [2016].

Our VR painting application enables the user to paint on 3D models with measured BRDFs. Our users praised how our application is intuitive to use, and how it creates a bridge between the analog and digital skills. Furthermore, it enables a fast and intuitive way to inspect BRDFs under various lighting conditions.

Acknowledgements

Paint bucket, table, and light bulb models are courtesy of TurboSquid.com. Environment maps: studio, garden, and sunset forest from hdrmaps.com, lobby and night city square from Joost Vanhoutte. We use chrome, blue paint, gold paint and red fabric BRDFs from the MERL database. Ogre model courtesy of Keenan Crane. VIVE and related assets are property of HTC, Inc. and Valve Corporation. We would like to thank users Felicia Frisvad, Jon Murray Vinther, Sam Surplice, Christian Ahm for their valuable feedback.

REFERENCES

M. Colbert, S. Pattanaik, and J. Krivanek. 2006. BRDF-Shop: creating physically correct bidirectional reflectance distribution functions. *IEEE Computer Graphics and Applications* 26, 1 (Jan 2006), 30–36. https://doi.org/10.1109/MCG.2006.13

B. Jackson and D. F. Keefe. 2016. Lift-off: Using reference imagery and freehand sketching to create 3D models in VR. *IEEE Transactions on Visualization and Computer Graphics* 22, 4 (2016), 1442–1451. https://doi.org/10.1109/TVCG.2016.2518099

W. Matusik, H. Pfister, M. Brand, and L. McMillan. 2003. A Data-Driven Reflectance Model. *ACM Transactions on Graphics* 22, 3 (July 2003), 759–769.

I. Wald, A. Dietrich, C. Benthin, A. Efremov, T. Dahmen, J. Gunther, V. Havran, H. p. Seidel, and P. Slusallek. 2006. Applying Ray Tracing for Virtual Reality and Industrial Design. In *2006 IEEE Symposium on Interactive Ray Tracing*. 177–185. https://doi.org/10.1109/RT.2006.280229

Appendix VI

Derivation of standard and directional dipole quantities

Derivation of standard and directional dipole quantities

Alessandro Dal Corso
Technical University of Denmark

Jeppe Revall Frisvad
Technical University of Denmark

Thomas Kim Kjeldsen
The Alexandra Institute

March 2018

1 Integrating the radiative transfer equation

We start from the radiative transfer equation [1]:

$$(\nabla \cdot \vec{\omega})L(\mathbf{x},\vec{\omega}) = -\sigma_t L(\mathbf{x},\vec{\omega}) + \sigma_s \int_{4\pi} p(\vec{\omega}',\vec{\omega})L(x,\vec{\omega}')\ d\omega' + q(\mathbf{x},\vec{\omega})\,,$$

where L is radiance at the position $\mathbf{x}$ in the direction $\vec{\omega}$. The equation describes how the directional derivative of L depends on the scattering properties of the surrounding medium, where σ_t is the extinction coefficient, σ_s is the scattering coefficient, and p is the phase function. Finally, q is the emitted radiance in the medium per unit length that we move along a ray. If we then integrate over all directions $\vec{\omega}$, we get

$$\int_{4\pi} (\nabla \cdot \vec{\omega})L(\mathbf{x},\vec{\omega})\ d\omega = \int_{4\pi} -\sigma_t L(\mathbf{x},\vec{\omega})\ d\omega + \int_{4\pi} \sigma_s \int_{4\pi} p(\vec{\omega}',\vec{\omega})L(x,\vec{\omega}')\ d\omega'\ d\omega + \int_{4\pi} q(\mathbf{x},\vec{\omega})\ d\omega\,.$$

Rearranging, we obtain

$$\nabla \cdot \left(\int_{4\pi} \vec{\omega} L(\mathbf{x},\vec{\omega})\ d\omega \right) = -\sigma_t \int_{4\pi} L(\mathbf{x},\vec{\omega})\ d\omega + \sigma_s \int_{4\pi} \left(\int_{4\pi} p(\vec{\omega}',\vec{\omega})\ d\omega \right)\ L(x,\vec{\omega}')\ d\omega' + Q_0(\mathbf{x})\,,$$

where we used the regularity of the operators to switch divergence and integral operations on the left-hand side, and to switch the integrals on the right-hand side. The integral of the phase function is 1, since it is normalized, so by further simplifying and applying the definitions of fluence ϕ and vector irradiance $\mathbf{E}$, we obtain

$$\begin{aligned} \nabla \cdot \mathbf{E}(\mathbf{x}) &= -\sigma_t \phi(\mathbf{x}) + \sigma_s \int_{4\pi} L(x,\vec{\omega}')\ d\omega' + Q_0(\mathbf{x}) \\ \nabla \cdot \mathbf{E}(\mathbf{x}) &= -\sigma_t \phi(\mathbf{x}) + \sigma_s \phi(\mathbf{x}) + Q_0(\mathbf{x}) \\ \nabla \cdot \mathbf{E}(\mathbf{x}) &= -\sigma_a \phi(\mathbf{x}) + Q_0(\mathbf{x})\,, \end{aligned} \tag{1}$$

where we introduced the absorption coefficient $\sigma_a = \sigma_t - \sigma_s$. Our Equation 1 then corresponds to Equation 1 in the work of Jensen et al. [6].

2 The diffusion approximation

To get the diffusion approximation, we approximate radiance using a second order spherical harmonics expansion:

$$L(\mathbf{x},\vec{\omega}) \approx \sum_{n=0}^{1}\sum_{m=-n}^{n} L_{n,m}(\mathbf{x})Y_{n,m}(\vec{\omega})\,,$$

where $Y_{n,m}(\vec{\omega})$ are normalized real-valued spherical harmonics basis functions, and $L_{n,m}(\mathbf{x})$ is the projection of L against the n,m basis function:

$$L_{n,m}(\mathbf{x}) = \int_{4\pi} L(\mathbf{x},\vec{\omega})Y_{n,m}(\vec{\omega})\;d\omega\,.$$

For $n=0$, the integral is trivial. The first term of the sum becomes:

$$L_{0,0}(\mathbf{x})Y_{0,0}(\vec{\omega}) = \int_{4\pi}\sqrt{\frac{1}{4\pi}}L(\mathbf{x},\vec{\omega})\;d\omega\sqrt{\frac{1}{4\pi}} = \frac{1}{4\pi}\phi(\mathbf{x})\,.$$

As for the other basis functions, we use the Cartesian form. We have

$$L_{1,-1}(\mathbf{x})Y_{1,-1}(\vec{\omega}) = \omega_x\int_{4\pi}\sqrt{\frac{3}{4\pi}}\omega_x L(\mathbf{x},\vec{\omega})\;d\omega\sqrt{\frac{3}{4\pi}} = \frac{3}{4\pi}\omega_x E_x(\mathbf{x})\,,$$

where the x subscript indicates the first component. Similarly, we obtain

$$L_{1,0}(\mathbf{x})Y_{1,0}(\vec{\omega}) = \frac{3}{4\pi}\omega_z E_z(\mathbf{x})$$

$$L_{1,1}(\mathbf{x})Y_{1,1}(\vec{\omega}) = \frac{3}{4\pi}\omega_y E_y(\mathbf{x})\,.$$

By applying the approximation, we finally obtain:

$$L(\mathbf{x},\vec{\omega}) \approx \frac{1}{4\pi}\phi(\mathbf{x})+\frac{3}{4\pi}\omega_x E_x(\mathbf{x})+\frac{3}{4\pi}\omega_y E_y(\mathbf{x})+\frac{3}{4\pi}\omega_z E_z(\mathbf{x}) = \frac{\phi(\mathbf{x})}{4\pi}+\frac{3}{4\pi}\vec{\omega}\cdot\mathbf{E}(\mathbf{x})\,.$$

3 The diffusion equation

To find an equation for the relation between the fluence and the optical properties of the medium, we substitute the diffusion approximation into the radiative transfer equation:

$$\begin{aligned}(\nabla\cdot\vec{\omega})\left(\frac{\phi(\mathbf{x})}{4\pi}+\frac{3}{4\pi}\vec{\omega}\cdot\mathbf{E}(\mathbf{x})\right) &= -\sigma_t\left(\frac{\phi(\mathbf{x})}{4\pi}+\frac{3}{4\pi}\vec{\omega}\cdot\mathbf{E}(\mathbf{x})\right)\\ &\quad+\sigma_s\int_{4\pi}p(\vec{\omega}',\vec{\omega})\left(\frac{\phi(\mathbf{x})}{4\pi}+\frac{3}{4\pi}\vec{\omega}'\cdot\mathbf{E}(\mathbf{x})\right)\;d\omega'+q(\mathbf{x},\vec{\omega})\,.\end{aligned}$$

For simplification, we need the following three identities:

$$\int_{4\pi}\vec{\omega}\;d\omega = 0$$

$$\int_{4\pi}\vec{\omega}(\vec{\omega}\cdot\mathbf{A})\;d\omega = \frac{4\pi}{3}\mathbf{A} \qquad (2)$$

$$\int_{4\pi} \vec{\omega}[\vec{\omega} \cdot \nabla(\vec{\omega} \cdot \mathbf{A})] \, d\omega = 0$$

We first multiply into parentheses in the equation above:

$$\frac{1}{4\pi}\vec{\omega}\cdot\nabla\phi(\mathbf{x})+\frac{3}{4\pi}\vec{\omega}\cdot\nabla(\vec{\omega}\cdot\mathbf{E}(\mathbf{x})) = -\sigma_t\frac{\phi(\mathbf{x})}{4\pi}-\sigma_t\frac{3}{4\pi}\vec{\omega}\cdot\mathbf{E}(\mathbf{x})+\sigma_s\frac{\phi(\mathbf{x})}{4\pi}\int_{4\pi} p(\vec{\omega}',\vec{\omega})\, d\omega' + \frac{3}{4\pi}\sigma_s\int_{4\pi} p(\vec{\omega}',\vec{\omega})\vec{\omega}'\cdot\mathbf{E}(\mathbf{x})\; d\omega' + q(\mathbf{x},\vec{\omega})\,.$$

Now, we multiply each term by $\vec{\omega}$ and integrate over the sphere. Taking all the terms separately, we have

$$\int_{4\pi} \frac{1}{4\pi}\vec{\omega}\cdot\nabla\phi(\mathbf{x})\vec{\omega}\; d\omega = \frac{1}{4\pi}\frac{4\pi}{3}\nabla\phi(\mathbf{x}) = \frac{\nabla\phi(\mathbf{x})}{3}$$

$$\int_{4\pi} \frac{3}{4\pi}\vec{\omega}\cdot\nabla(\vec{\omega}\cdot\mathbf{E}(\mathbf{x}))\vec{\omega}\; d\omega = \frac{3}{4\pi}\int_{4\pi}\vec{\omega}[\vec{\omega}\cdot\nabla(\vec{\omega}\cdot\mathbf{E}(\mathbf{x}))]\; d\omega = 0$$

$$\int_{4\pi} -\sigma_t\frac{\phi(\mathbf{x})}{4\pi}\vec{\omega}\; d\omega = -\sigma_t\frac{\phi(\mathbf{x})}{4\pi}\int_{4\pi}\vec{\omega}\; d\omega = 0$$

$$\int_{4\pi} -\sigma_t\frac{3}{4\pi}\vec{\omega}\cdot\mathbf{E}(\mathbf{x})\vec{\omega}\; d\omega = -\sigma_t\frac{3}{4\pi}\int_{4\pi}\vec{\omega}(\vec{\omega}\cdot\mathbf{E}(\mathbf{x}))\; d\omega = -\sigma_t\frac{3}{4\pi}\frac{4\pi}{3}\mathbf{E}(\mathbf{x}) = -\sigma_t\mathbf{E}(\mathbf{x})$$

$$\int_{4\pi}\sigma_s\frac{\phi(\mathbf{x})}{4\pi}\int_{4\pi} p(\vec{\omega}',\vec{\omega})\; d\omega'\vec{\omega}\; d\omega = \sigma_s\frac{\phi(\mathbf{x})}{4\pi}\int_{4\pi}\vec{\omega}\; d\omega = 0$$

$$\int_{4\pi}\frac{3}{4\pi}\sigma_s\int_{4\pi} p(\vec{\omega}',\vec{\omega})\vec{\omega}'\cdot\mathbf{E}(\mathbf{x})\; d\omega'\vec{\omega}\; d\omega \overset{(*)}{=} g\sigma_s\mathbf{E}(\mathbf{x})$$

$$\int_{4\pi} q(\mathbf{x},\vec{\omega})\vec{\omega}\; d\omega = \mathbf{Q}_1(\mathbf{x})\,,$$

Passage $(*)$ is a bit more delicate as it requires the assumption that the phase function is rotationally symmetric, which means that it depends only on the cosine between the two direction vector arguments ($p(\vec{\omega}',\vec{\omega}) = p(\vec{\omega}'\cdot\vec{\omega})$). For the interested reader, we further discuss this result at the end of this section (Section 3.1). Putting everything together:

$$\frac{\nabla\phi(\mathbf{x})}{3} + 0 = 0 - \sigma_t\mathbf{E}(\mathbf{x}) + 0 + g\sigma_s\mathbf{E}(\mathbf{x}) + \mathbf{Q}_1(\mathbf{x})$$

$$\nabla\phi(\mathbf{x}) = -3\sigma_t'\mathbf{E}(\mathbf{x}) + 3\mathbf{Q}_1(\mathbf{x})\,, \tag{3}$$

where we used $\sigma_t' = \sigma_s' + \sigma_a = \sigma_s(1-g)+\sigma_a = \sigma_t - g\sigma_s$. To obtain the diffusion equation, we need to combine Equations 3 and 1. We first rearrange Equation 3:

$$\mathbf{E}(\mathbf{x}) = 3D\mathbf{Q}_1(\mathbf{x}) - D\nabla\phi(\mathbf{x})\,,$$

where $D = \frac{1}{3\sigma_t'}$. Inserting into Equation 1 (and assuming a homogeneous medium), we have

$$\nabla\cdot(3D\mathbf{Q}_1(\mathbf{x}) - D\nabla\phi(\mathbf{x})) = -\sigma_a\phi(\mathbf{x}) + Q_0(\mathbf{x})$$

$$3D\nabla\cdot\mathbf{Q}_1(\mathbf{x}) - D\nabla^2\phi(\mathbf{x}) = -\sigma_a\phi(\mathbf{x}) + Q_0(\mathbf{x})$$

$$D\nabla^2\phi(\mathbf{x}) = \sigma_a\phi(\mathbf{x}) - Q_0(\mathbf{x}) + 3D\nabla\cdot\mathbf{Q}_1(\mathbf{x})\,, \tag{4}$$

which is the diffusion equation as it appears in the work of Jensen et al. [6].

3.1 Rotationally symmetric phase function

Assuming that p is rotationally symmetric so that $p(\vec{\omega}', \vec{\omega}) = p(\vec{\omega}' \cdot \vec{\omega})$, we can expand it in Legendre polynomials:

$$p(\vec{\omega}' \cdot \vec{\omega}) = \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} p_n P_n(\vec{\omega}' \cdot \vec{\omega}) \,,$$

where p_n are the expansion coefficients and P_n are the Legendre polynomials. Since

$$P_0(\vec{\omega}' \cdot \vec{\omega}) = 1, \quad P_1(\vec{\omega}' \cdot \vec{\omega}) = \vec{\omega}' \cdot \vec{\omega} \,,$$

we have

$$p_0 = \int_{4\pi} p(\vec{\omega}' \cdot \vec{\omega}) P_0(\vec{\omega}' \cdot \vec{\omega}) \; d\omega = 1 \,,$$

$$p_1 = \int_{4\pi} p(\vec{\omega}' \cdot \vec{\omega}) P_1(\vec{\omega}' \cdot \vec{\omega}) \; d\omega = \int_{4\pi} p(\vec{\omega}' \cdot \vec{\omega})(\vec{\omega}' \cdot \vec{\omega}) \; d\omega = g \,.$$

In addition, the orthogonality relations for Legendre polynomials are

$$\int_{4\pi} P_n(\vec{\omega}' \cdot \vec{\omega}) P_m(\vec{\omega}' \cdot \vec{\omega}) \; d\omega' = \begin{cases} \dfrac{4\pi}{2n+1} & , \quad \text{for } n = m \\ 0 & , \quad \text{otherwise} \,. \end{cases}$$

Noting that in spherical harmonics with $\vec{\omega}$ as the local z-axis:

$$\begin{aligned} \omega'_x &= -\sqrt{\frac{4\pi}{3}} Y_{1,1}(\vec{\omega}') \\ \omega'_y &= -\sqrt{\frac{4\pi}{3}} Y_{1,-1}(\vec{\omega}') \\ \omega'_z &= \sqrt{\frac{4\pi}{3}} Y_{1,0}(\vec{\omega}') = P_1(\vec{\omega}' \cdot \vec{\omega}) \,, \end{aligned}$$

and that integration over ω'_x and ω'_y are zero, the expansion of the phase function combined with the orthogonality of the Legendre polynomials leave us with

$$\begin{aligned} &\int_{4\pi} \frac{3}{4\pi} \sigma_s \int_{4\pi} p(\vec{\omega}', \vec{\omega}) \vec{\omega}' \cdot \mathbf{E}(\mathbf{x}) \; d\omega' \vec{\omega} \; d\omega \\ &= \int_{4\pi} \frac{3}{4\pi} \sigma_s \frac{3}{4\pi} g \left(\frac{4\pi}{3} \vec{\omega} \cdot \mathbf{E}(\mathbf{x}) \right) \vec{\omega} \; d\omega = g \sigma_s \mathbf{E}(\mathbf{x}) \,, \end{aligned}$$

which is the expected result.

4 Boundary condition

In the case of a scattering medium in a half-space, we impose the classic boundary condition that the net inward flux on each surface point $\mathbf{x}_s$ with (inward) normal $\vec{n}_s$ is zero:

$$\int_{2\pi_+} L(\mathbf{x}_s, \vec{\omega})(\vec{\omega} \cdot \vec{n}_s) \; d\omega = 0 \,.$$

We use the diffusion approximation:

$$\int_{2\pi_+} \left(\frac{\phi(\mathbf{x_s})}{4\pi} + \frac{3}{4\pi} \vec{\omega} \cdot \mathbf{E}(\mathbf{x_s}) \right) (\vec{\omega} \cdot \vec{n}_s) \, d\omega = 0$$

$$\phi(\mathbf{x}) \int_{2\pi_+} (\vec{\omega} \cdot \vec{n}_s) \, d\omega + 3 \int_{2\pi_+} (\vec{\omega} \cdot \mathbf{E}(\mathbf{x_s}))(\vec{n}_s \cdot \vec{\omega}) \, d\omega = 0 \, .$$

Given the standard spherical coordinates convention, $n_s = (0, 0, 1)$ and $\vec{\omega} = (\cos\phi \sin\theta, \sin\phi \sin\theta, \cos\theta)$. We then obtain

$$\int_{2\pi_+} (\vec{\omega} \cdot \vec{n}_s) \, d\omega = \int_0^{2\pi} \int_0^{\frac{\pi}{2}} \cos\theta \sin\theta d\theta d\phi = \pi \, ,$$

and

$$\int_{2\pi_+} (\vec{\omega} \cdot \mathbf{E}(\mathbf{x}))(\vec{n}_s \cdot \vec{\omega}) \, d\omega = \int_0^{2\pi} \int_0^{\frac{\pi}{2}} (\cos\phi \sin\theta E_x + \sin\phi \sin\theta E_y + \cos\theta E_z) \cos\theta \sin\theta d\theta d\phi$$

$$= \frac{2\pi}{3} E_z = \frac{2\pi}{3} \vec{n}_s \cdot \mathbf{E}(\mathbf{x_s}) \, .$$

Using the last two results and simplifying, we get:

$$\phi(\mathbf{x})\pi + 3 \left(\frac{2\pi}{3} \vec{n}_s \cdot \mathbf{E}(\mathbf{x_s}) \right) = 0$$

$$\phi(\mathbf{x}) + 2\vec{n}_s \cdot \mathbf{E}(\mathbf{x_s}) = 0 \, .$$

From Equation 3, assuming no emission in $\mathbf{x}_s$, we have $\mathbf{Q}_1(\mathbf{x_s}) = \mathbf{0}$, so

$$\mathbf{E}(\mathbf{x_s}) = -D \nabla \phi(\mathbf{x_s}) \, .$$

Inserting this result and simplifying, we get the final boundary condition:

$$\phi(\mathbf{x_s}) - 2D(\vec{n}_s \cdot \nabla)\phi(\mathbf{x_s}) = 0 \, .$$

5 Different media assumption

To include nonzero inward flux at boundaries, we need to change the above equations. The boundary condition then becomes:

$$I_+ = \int_{2\pi_+} L(\mathbf{x}_s, \vec{\omega})(\vec{\omega} \cdot \vec{n}_s) \, d\omega = \int_{2\pi_-} R(\eta, \vec{\omega}) L(\mathbf{x}_s, \vec{\omega})(-\vec{\omega} \cdot \vec{n}_s) \, d\omega = I_- \, . \quad (5)$$

Keeping the above conventions, we define the Fresnel reflectance R by:

$$R(\eta, \vec{\omega}) = \begin{cases} 1 & \text{for } \frac{\pi}{2} \leq \theta \leq \pi - \theta_c \\ F_r(\eta, \vec{\omega} \cdot \vec{n}_s) & \text{for } \pi - \theta_c \leq \theta \leq \pi \, , \end{cases}$$

where θ_c is the critical angle, and

$$F_r(\eta, \mu) = \frac{1}{2} \left[\left(\frac{\mu - \eta\mu_0}{\mu + \eta\mu_0} \right)^2 + \left(\frac{\eta\mu - \mu_0}{\eta\mu + \mu_0} \right)^2 \right] \quad \text{with} \quad \mu_0^2 = 1 - \eta^2(1 - \mu^2)$$

and $\cos^2\theta_c = \max(1-\eta^{-2}, 0)$. In principle, if we allow complex numbers, we would have $R(\eta, \vec{\omega}) = F_r(\eta, \vec{\omega}\cdot\vec{n}_s)$.

The left side of Equation 5 is:

$$I_+ = \frac{1}{4}(\phi(\mathbf{x}) - 2D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}))\,.$$

The other side is more tricky, since it requires splitting the integration along the different angles. We proceed as before, introducing the diffusion approximation:

$$I_- = \frac{\phi(\mathbf{x})}{4\pi}\int_{2\pi_-} R(\eta,\vec{\omega})(-\vec{\omega}\cdot\vec{n}_s)\;d\omega + \frac{3}{4\pi}\int_{2\pi_-} R(\eta,\vec{\omega})(\vec{\omega}\cdot\mathbf{E}(\mathbf{x}))(-\vec{\omega}\cdot\vec{n}_s)\;d\omega\,.$$

The cosine-weighted integration of the Fresnel reflectance is sometimes referred to as diffuse Fresnel reflectance F_{dr}. If we, outside the region of total internal reflection, approximate the R function by Fresnel reflectance for normal incidence $R_0 = F_r(\eta, 1)$, we can find an approximate analytical solution. The first part is then:

$$\begin{aligned}\int_{2\pi_-} R(\eta,\vec{\omega})(-\vec{\omega}\cdot\vec{n}_s)\,d\omega &= \int_0^{2\pi}\int_{\frac{\pi}{2}}^{\pi-\theta_c}(-\cos\theta)\sin\theta d\theta d\phi + \int_0^{2\pi}\int_{\pi-\theta_c}^{\pi} R_0(-\cos\theta)\sin\theta d\theta d\phi\\ &= \pi((1-R_0)\cos^2\theta_c + R_0)\,.\end{aligned}$$

The second part (only on the z-coordinate, since the other coordinates are zero):

$$\begin{aligned}&\int_{2\pi_-} R(\eta,\vec{\omega})(\vec{\omega}\cdot\mathbf{E}(\mathbf{x}))(-\vec{\omega}\cdot\vec{n}_s)\;d\omega\\ &= E_z\int_0^{2\pi}\int_{\frac{\pi}{2}}^{\pi-\theta_c}(-\cos^2\theta)\sin\theta d\theta d\phi + E_z\int_0^{2\pi}\int_{\pi-\theta_c}^{\pi} R_0(-\cos^2\theta)\sin\theta d\theta d\phi\\ &= \frac{2\pi}{3}E_z(R_0(\cos^3\theta_c - 1) - \cos^3\theta_c)\,.\end{aligned}$$

Performing all simplifications, we finally get:

$$I_- = \frac{1}{4}[((1-R_0)\cos^2\theta_c + R_0)\phi(x) - 2D(R_0(\cos^3\theta_c - 1) - \cos^3\theta_c)(\vec{n}_s\cdot\nabla)\phi(\mathbf{x})]\,.$$

We can now impose $I_+ = I_-$:

$$\phi(\mathbf{x}) - 2D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) = ((1-R_0)\cos^2\theta_c + R_0)\phi(x) - 2D(R_0(\cos^3\theta_c - 1) - \cos^3\theta_c)(\vec{n}_s\cdot\nabla)\phi(\mathbf{x})\,,$$

which we can simplify as follows:

$$\begin{aligned}\phi(\mathbf{x}) - 2\frac{1+R_0+(1-R_0)\cos^3\theta_c}{1-R_0-(1-R_0)\cos^2\theta_c}D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) &= 0\\ \phi(\mathbf{x}) - 2\frac{\frac{1+R_0}{1-R_0}+\cos^3\theta_c}{1-\cos^2\theta_c}D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) &= 0\\ \phi(\mathbf{x}) - 2AD(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) &= 0\,.\end{aligned}$$

So, to handle reflective boundaries, we need to add a correction factor A in our boundary condition. With our current approximation, we have

$$A = \frac{\frac{1+R_0}{1-R_0}+\cos^3\theta_c}{1-\cos^2\theta_c} = \frac{\frac{\eta^2+1}{2\eta}+[\max(1-\eta^{-2},0)]^{\frac{3}{2}}}{1-\max(1-\eta^{-2},0)}\,.$$

5.1 Approximating the corrective factor

Assuming separability, we can rewrite the I_- term in Equation 5 as:

$$\int_{2\pi_-} R(\eta,\vec{\omega})L(x_s,\vec{\omega})(-\vec{\omega}\cdot\vec{n}_s)\,d\omega \approx \int_{2\pi_-} R(\eta,\vec{\omega})(-\vec{\omega}\cdot\vec{n}_s)\,d\omega \int_{2\pi_-} L(x_s,\vec{\omega})(-\vec{\omega}\cdot\vec{n}_s)\,d\omega$$

$$= F_{dr}(\eta)\ \frac{1}{4}(\phi(\mathbf{x}) + 2D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}))$$

An approximate fit of the $F_{dr}(\eta)$ integral is [3]

$$F_{dr}(\eta) = \begin{cases} -0.4399 + \dfrac{0.7099}{\eta} - \dfrac{0.3319}{\eta^2} + \dfrac{0.0636}{\eta^3} & , \quad \eta < 1 \\ -\dfrac{1.4399}{\eta^2} + \dfrac{0.7099}{\eta} + 0.6681 + 0.0636\eta & , \quad \eta > 1\,. \end{cases}$$

So we can express the boundary condition as

$$\phi(\mathbf{x}) - 2\pi D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) = F_{dr}(\eta)\ (\phi(\mathbf{x}) + 2D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}))$$

$$\phi(\mathbf{x})(1 - F_{dr}) - 2D(1 + F_{dr})(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) = 0$$

$$\phi(\mathbf{x}) - 2D\frac{1 + F_{dr}}{1 - F_{dr}}(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) = 0$$

$$\phi(\mathbf{x}) - 2AD(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) = 0$$

with $A = \frac{1+F_{dr}}{1-F_{dr}}$, which returns values fairly close to the A found in the previous section. The A in this section is the one employed by Jensen et al. [6]. With respect to F_{dr}, they only provide the more common case of $\eta > 1$.

6 Solutions for an infinite medium

From the diffusion equation (4), we have

$$(D\nabla^2 - \sigma_a)\phi(\mathbf{x}) = -Q_0(\mathbf{x}) + 3D\nabla\cdot\mathbf{Q}_1(\mathbf{x})$$

$$(\nabla^2 - \sigma_{tr}^2)\phi(\mathbf{x}) = -\frac{Q_0(\mathbf{x})}{D} + 3\nabla\cdot\mathbf{Q}_1(\mathbf{x})\,,$$

which is a particular case of the screened Poisson equation. This has a generic solution based on the method of Green's functions:

$$\phi(\mathbf{x}) = \frac{1}{4\pi}\iiint_{\mathbb{R}^3} \frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\left(\frac{Q_0(\mathbf{r}')}{D} - 3\nabla\cdot\mathbf{Q}_1(\mathbf{r}')\right)d^3\mathbf{r}'\,.$$

6.1 Point source solutions

If we use a point source placed at the origin, we have

$$Q_0(\mathbf{x}) = \Phi_i\delta(\mathbf{x})$$

$$\mathbf{Q}_1(\mathbf{x}) = 0\,.$$

Inserting in the solution based on Green's function:

$$\phi(\mathbf{x}) = \frac{1}{4\pi} \iiint_{\mathbb{R}^3} \frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|} \left(\frac{\Phi_i \delta(\mathbf{r}')}{D} \right) d^3\mathbf{r}'$$

and applying the delta function, the result is

$$\phi(\mathbf{x}) = \frac{\Phi_i}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r} \,,$$

where $r = \|\mathbf{x}\|$ is the distance to the point of interest. A similar result is obtained if we consider a ray source at the origin with direction along the inward surface normal (z-axis). Suppose the medium exhibits isotropic scattering, then the source of first scattering events is [10]

$$Q_0(\mathbf{x}) = \Phi_i \sigma_s' \delta(x)\delta(y)\Theta(z) e^{-\sigma_t' z}$$

$$\mathbf{Q}_1(\mathbf{x}) = 0 \,,$$

where $\Theta(z)$ is the Heaviside step function, which is 1 for $z \geq 0$ and 0 otherwise. Inserting in the solution based on Green's function with $\mathbf{r}' = (x', y', z')$:

$$\phi(\mathbf{x}) = \frac{1}{4\pi} \iiint_{\mathbb{R}^3} \frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|} \left(\frac{\Phi_i \sigma_s' \delta(x')\delta(y')\Theta(z') e^{-\sigma_t' z'}}{D} \right) d^3\mathbf{r}'$$

and applying the deltas:

$$\phi(\mathbf{x}) = \frac{\Phi_i}{4\pi D} \int_0^{+\infty} \frac{e^{-\sigma_{tr}\|\mathbf{x}-z'\vec{n}_s\|}}{\|\mathbf{x}-z'\vec{n}_s\|} \left(\sigma_s' e^{-\sigma_t' z'} \right) dz' \,.$$

Considering positions $\mathbf{x}$ in the xy-plane far from the origin and the exponential attenuation with increasing z', we use the assumption $\|\mathbf{x} - z'\vec{n}_s\| \approx \|\mathbf{x}\| = r$ and get

$$\phi(\mathbf{x}) = \frac{\Phi_i}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r} \sigma_s' \int_0^{+\infty} e^{-\sigma_t' z'} dz' = \frac{\Phi_i}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r} \frac{\sigma_s'}{\sigma_t'} = \alpha' \frac{\Phi_i}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r} \,,$$

where $\alpha' = \frac{\sigma_s'}{\sigma_t'}$ is the reduced scattering albedo. Thus, we get the monopole solution for a ray source of normal incidence:

$$\phi(\mathbf{x}) = \frac{\Phi}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r} \qquad \text{with } \Phi = \alpha' \Phi_i. \tag{6}$$

6.2 Ray source solution

In case of a ray source that is not along the normal direction and not necessarily in an isotropic medium, we can use the following equations for the source terms [8]:

$$Q_0(\mathbf{x}) = \Phi_i \tilde{\sigma}_s \delta(x)\delta(y)\Theta(z) e^{-\tilde{\sigma}_t z}$$

$$\mathbf{Q}_1(\mathbf{x}) = \tilde{g} Q_0(\mathbf{x}) \vec{n}_s \,,$$

where we have used the delta-Eddington scattering properties [7]:

$$\tilde{\sigma}_s = \sigma_s(1 - g^2) \quad , \quad \tilde{\sigma}_t = \tilde{\sigma}_s + \sigma_a \quad , \quad \tilde{g} = g/(g+1) \,.$$

Inserting in the diffusion equation (4), we get two integrals when using the Green's function solution. Splitting the solution accordingly: $\phi(\mathbf{x}) = \phi_1(\mathbf{x}) + \phi_2(\mathbf{x})$, we have

$$\begin{aligned}\phi_1(\mathbf{x}) &= \frac{1}{4\pi}\iiint_{\mathbb{R}^3}\frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\left(\frac{\Phi_i\tilde{\sigma}_s\delta(x')\delta(y')\Theta(z')e^{-\tilde{\sigma}_t z'}}{D}\right)d^3\mathbf{r}' \\ &= \frac{\Phi_i\tilde{\sigma}_s}{4\pi D}\int_0^{+\infty}\frac{e^{-\sigma_{tr}\|\mathbf{x}-z'\vec{n}_s\|}}{\|\mathbf{x}-z'\vec{n}_s\|}e^{-\tilde{\sigma}_t z'}dz' \\ &= \frac{3\tilde{\sigma}_s\Phi_i}{4\pi}\tilde{\sigma}_t\int_0^{+\infty}\frac{e^{-\sigma_{tr}\|\mathbf{x}-z'\vec{n}_s\|}}{\|\mathbf{x}-z'\vec{n}_s\|}e^{-\tilde{\sigma}_t z'}dz'\end{aligned}$$

and

$$\phi_2(\mathbf{x}) = -\frac{3}{4\pi}\iiint_{\mathbb{R}^3}\frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\nabla\cdot\left(\Phi_i\tilde{\sigma}_s\tilde{g}\delta(x')\delta(y')\Theta(z')e^{-\tilde{\sigma}_t z'}\right)\vec{n}_s\,d^3\mathbf{r}'\,.$$

We now apply the divergence operator:

$$\phi_2(\mathbf{x}) = -\frac{3\Phi_i\tilde{\sigma}_s\tilde{g}}{4\pi}\iiint_{\mathbb{R}^3}\frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\left[\delta(x)\delta(y)\frac{\partial}{\partial z'}(\Theta(z)e^{-\tilde{\sigma}_t z'})\right]d^3\mathbf{r}'.$$

Note that we have only the z-term given that we multiply by $\vec{n}_s = (0,0,1)$. Using that $\frac{\partial\Theta(z')}{\partial z'} = \delta(z')$, we have

$$\phi_2(\mathbf{x}) = -\frac{3\Phi_i\tilde{\sigma}_s\tilde{g}}{4\pi}\iiint_{\mathbb{R}^3}\frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\delta(x')\delta(y')\left[\delta(z')e^{-\tilde{\sigma}_t z'} - \tilde{\sigma}_t\Theta(z')e^{-\tilde{\sigma}_t z'}\right]d^3\mathbf{r}'.$$

Applying the deltas, we get

$$\begin{aligned}\phi_2(\mathbf{x}) &= -\frac{3\Phi_i\tilde{\sigma}_s\tilde{g}}{4\pi}\frac{e^{-\sigma_{tr}r}}{r} + \frac{3\Phi_i\tilde{\sigma}_s\tilde{g}\tilde{\sigma}_t}{4\pi}\iiint_{\mathbb{R}^3}\frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\delta(x')\delta(y')\Theta(z')e^{-\tilde{\sigma}_t z'}d^3\mathbf{r}' \\ &= -\frac{3\Phi_i\tilde{\sigma}_s\tilde{g}}{4\pi}\frac{e^{-\sigma_{tr}r}}{r} + \frac{3\Phi_i\tilde{\sigma}_s\tilde{g}\tilde{\sigma}_t}{4\pi}\int_0^{+\infty}\frac{e^{-\sigma_{tr}\|\mathbf{x}-z'\vec{n}_s\|}}{\|\mathbf{x}-z'\vec{n}_s\|}e^{-\tilde{\sigma}_t z'}dz'\,.\end{aligned}$$

Putting it together:

$$\begin{aligned}\phi(\mathbf{x}) &= -\frac{3\Phi_i\tilde{\sigma}_s\tilde{g}}{4\pi}\frac{e^{-\sigma_{tr}r}}{r} + \frac{3\tilde{\sigma}_s\Phi_i}{4\pi}(\tilde{\sigma}_t + \tilde{\sigma}_s\tilde{g} + \sigma_a\tilde{g})\int_0^{+\infty}\frac{e^{-\sigma_{tr}\|\mathbf{x}-z'\vec{n}_s\|}}{\|\mathbf{x}-z'\vec{n}_s\|}e^{-\tilde{\sigma}_t z'}dz' \\ \phi(\mathbf{x}) &= \frac{3\Phi_i\tilde{\sigma}_s}{4\pi}\left(-\tilde{g}\frac{e^{-\sigma_{tr}r}}{r} + (\tilde{\sigma}_s + \sigma_a(1+\tilde{g}))\int_0^{+\infty}\frac{e^{-\sigma_{tr}\|\mathbf{x}-z'\vec{n}_s\|}}{\|\mathbf{x}-z'\vec{n}_s\|}e^{-\tilde{\sigma}_t z'}dz'\right).\end{aligned} \tag{7}$$

We can interpret the second term in Eq. 7 as the fluence from an exponentially decaying line source along the z-axis. Due to this exponentially decaying factor, the integrand will only have a significant weight for small z. Hence, in the asymptotic limit, $r \gg 1/\tilde{\sigma}_s$, we can approximate the distances in the integrand

$$\begin{aligned}\|\mathbf{x}-z'\vec{n}_s\| &= \sqrt{r^2+z'^2-2z'r\cos\theta} = r\left(1-2\frac{z'}{r}\cos\theta+\frac{z'^2}{r^2}\right)^{\frac{1}{2}} \\ &\approx r\left(1-\frac{z'}{r}\cos\theta\right) = r - z'\cos\theta\,,\end{aligned}$$

and

$$\begin{aligned}\frac{1}{\|\mathbf{x}-z'\vec{n}_s\|} &= \frac{1}{\sqrt{r^2+z'^2-2z'r\cos\theta}} = \frac{1}{r}\left(1-2\frac{z'}{r}\cos\theta+\frac{z'^2}{r^2}\right)^{-\frac{1}{2}} \\ &\approx \frac{1}{r}\left(1+\frac{z'}{r}\cos\theta\right).\end{aligned}$$

Then

$$\begin{aligned}\int_0^\infty \frac{e^{-\tilde{\sigma}_t z'}e^{-\sigma_{tr}\|\mathbf{x}-z'\vec{n}_s\|}}{\|\mathbf{x}-z'\vec{n}_s\|}dz' &\approx \frac{e^{-\sigma_{tr}r}}{r}\int_0^\infty e^{-(\tilde{\sigma}_t-\sigma_{tr}\cos\theta)z'}\left(1+\frac{z'}{r}\cos\theta\right)dz' \\ &= \frac{e^{-\sigma_{tr}r}}{r}\left(\frac{1}{\tilde{\sigma}_t-\sigma_{tr}\cos\theta}+\frac{\cos\theta}{r}\frac{1}{(\tilde{\sigma}_t-\sigma_{tr}\cos\theta)^2}\right).\end{aligned}$$

In a highly scattering medium, $\sigma_a \ll \tilde{\sigma}_s$. For $g \neq 1$ and $\sigma_a \ll \sigma_{tr} \ll \tilde{\sigma}_s$, this will imply

$$\frac{1}{\tilde{\sigma}_t-\sigma_{tr}\cos\theta} = \frac{1}{\tilde{\sigma}_s}\left(1+\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}\right)^{-1} \approx \frac{1}{\tilde{\sigma}_s}\left(1-\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}\right)$$

and

$$\frac{1}{(\tilde{\sigma}_t-\sigma_{tr}\cos\theta)^2} = \frac{1}{\tilde{\sigma}_s^2}\left(1+\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}\right)^{-2} \approx \frac{1}{\tilde{\sigma}_s^2}\left(1-2\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}\right).$$

The integral can now be approximated by

$$\begin{aligned}\int_0^\infty \frac{e^{-\tilde{\sigma}_t z'}e^{-\sigma_{tr}\|\mathbf{x}-z'\vec{n}_s\|}}{\|\mathbf{x}-z'\vec{n}_s\|}dz' &\approx \frac{e^{-\sigma_{tr}r}}{r}\left(\frac{1}{\tilde{\sigma}_s+\sigma_a-\sigma_{tr}\cos\theta}+\frac{\cos\theta}{r}\frac{1}{(\tilde{\sigma}_s+\sigma_a-\sigma_{tr}\cos\theta)^2}\right) \\ &\approx \frac{e^{-\sigma_{tr}r}}{\tilde{\sigma}_s r}\left(1-\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}+\frac{\cos\theta}{\tilde{\sigma}_s r}\left(1-2\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}\right)\right) \\ &= \frac{e^{-\sigma_{tr}r}}{\tilde{\sigma}_s r}\left(1-\frac{\sigma_a}{\tilde{\sigma}_s}+\frac{\cos\theta}{\tilde{\sigma}_s}\left(\sigma_{tr}+\frac{1}{r}-2\frac{\sigma_a}{\tilde{\sigma}_s r}\right)+2\frac{\sigma_{tr}\cos^2\theta}{\tilde{\sigma}_s^2 r}\right).\end{aligned}$$

Inserting this expression in Eq. 7,

$$\begin{aligned}\frac{\phi(\mathbf{x})}{\Phi_i} &\approx -\frac{3\tilde{g}\tilde{\sigma}_s e^{-\sigma_{tr}r}}{4\pi r} \\ &\quad +\frac{3\left(\tilde{\sigma}_s+(\tilde{g}+1)\sigma_a\right)e^{-\sigma_{tr}r}}{4\pi r}\left(1-\frac{\sigma_a}{\tilde{\sigma}_s}+\frac{\cos\theta}{\tilde{\sigma}_s}\left(\sigma_{tr}+\frac{1}{r}-2\frac{\sigma_a}{\tilde{\sigma}_s r}\right)+2\frac{\sigma_{tr}\cos^2\theta}{\tilde{\sigma}_s^2 r}\right) \\ &= \frac{3e^{-\sigma_{tr}r}}{4\pi r}\left(\left(\tilde{\sigma}_s+(\tilde{g}+1)\sigma_a\right)\left(1-\frac{\sigma_a}{\tilde{\sigma}_s}\right)-\tilde{g}\tilde{\sigma}_s\right) \\ &\quad +\frac{3e^{-\sigma_{tr}r}}{4\pi r}\cos\theta\left(1+(\tilde{g}+1)\frac{\sigma_a}{\tilde{\sigma}_s}\right)\left(\sigma_{tr}+\frac{1}{r}\left(1-2\frac{\sigma_a}{\tilde{\sigma}_s}\right)\right) \\ &\quad +\frac{3\left(\tilde{\sigma}_s+(\tilde{g}+1)\sigma_a\right)e^{-\sigma_{tr}r}}{4\pi r}2\frac{\sigma_{tr}\cos^2\theta}{\tilde{\sigma}_s^2 r}.\end{aligned}$$

Now we neglect terms $\sigma_a/\tilde{\sigma}_s$ compared to unity

$$\frac{\phi(\mathbf{x})}{\Phi_i} \approx \frac{3e^{-\sigma_{tr}r}}{4\pi r}\tilde{\sigma}_s\left(1-\tilde{g}\right)+\frac{3e^{-\sigma_{tr}r}}{4\pi r}\cos\theta\left(\sigma_{tr}+\frac{1}{r}\right)+\frac{3e^{-\sigma_{tr}r}}{4\pi r}2\frac{\sigma_{tr}\cos^2\theta}{\tilde{\sigma}_s r}.$$

Introducing the reduced scattering coefficient $\sigma'_s = \tilde{\sigma}_s(1-\tilde{g}) = \sigma_s(1-g)$, and neglecting terms $\sigma_{tr}/\tilde{\sigma}_s \cdot 1/(\tilde{\sigma}_s r)$ compared to unity,

$$\phi(\mathbf{x}) \approx \frac{3\Phi_i e^{-\sigma_{tr}r}}{4\pi r}\sigma'_s + \frac{3\Phi_i e^{-\sigma_{tr}r}}{4\pi r}\cos\theta\left(\sigma_{tr}+\frac{1}{r}\right) = \frac{\Phi_i}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r}\left(1+3D\frac{1+\sigma_{tr}r}{r}\cos\theta\right),$$

where $1/D$ is used in place of $3\sigma'_s$, since the σ'_s can be approximated by σ'_t when absorption is negligible compared to scattering. This is a valid assumption in highly scattering media.

7 Fresnel integrals

To aid in our calculations, we define the Fresnel transmittance integrals of the first two orders:

$$C_\phi(\eta) = \frac{1}{4\pi}\int_{2\pi} T_{21}(\eta,\theta_o)\cos\theta_o \mathrm{d}\vec{\omega}_o$$

$$C_{\mathbf{E}}(\eta) = \frac{3}{4\pi}\int_{2\pi} T_{21}(\eta,\theta_o)\cos^2\theta_o \mathrm{d}\vec{\omega}_o\,,$$

where $\cos\theta_o = \vec{n}_o\cdot\vec{\omega}_o$, and the integral is on the whole hemisphere where $\vec{n}_o\cdot\vec{\omega}_o > 0$. These integrals are similar to F_{dr}, but based on Fresnel transmittance instead of Fresnel reflectance.

8 BSSRDF theory

The BSSRDF is defined as the ratio of an element of emergent radiance L_r to an element of incident flux Φ_i [9]:

$$S(\mathbf{x}_i,\vec{\omega}_i,\mathbf{x}_o,\vec{\omega}_o) = \frac{\mathrm{d}L_r(\mathbf{x}_o,\vec{\omega}_o)}{\mathrm{d}\Phi_i(\mathbf{x}_i,\vec{\omega}_i)}\,.$$

There are various approximations available for the BSSRDF. For the directional dipole [5], the BSSRDF is split into the following terms:

$$S = T_{12}(S_{\delta E} + S_d)T_{21}\,.$$

Let us consider only the diffusive part, S_d. The emergent radiance due to diffusion events is given by

$$L_{r,d}(\mathbf{x}_o,\vec{\omega}_o) = \eta^2 T_{21} L_d(\mathbf{x}_o,\vec{\omega}_{21})\,,$$

where $\vec{\omega}_{21}$ is the refracted vector corresponding to $\vec{\omega}_o$:

$$\vec{\omega}_{21} = \frac{\vec{\omega}_o}{\eta} - \left(\frac{\vec{n}_o\cdot\vec{\omega}_o}{\eta} - \sqrt{1-\frac{1-(\vec{n}_o\cdot\vec{\omega}_o)^2}{\eta^2}}\right)\vec{n}_o\,.$$

Combining the above equations we obtain:

$$S_d(\mathbf{x}_i,\vec{\omega}_i,\mathbf{x}_o,\vec{\omega}_o) = T_{12}S_dT_{21} = \frac{\mathrm{d}L_{r,d}(\mathbf{x}_o,\vec{\omega}_o)}{\mathrm{d}\Phi_i(\mathbf{x}_i,\vec{\omega}_i)} = \eta^2\frac{\mathrm{d}T_{21}L_d(\mathbf{x}_o,\vec{\omega}_{21})}{\mathrm{d}\Phi_i(\mathbf{x}_i,\vec{\omega}_i)}$$

We now integrate over the cosine-weighted hemisphere with $\vec{n}_o \cdot \vec{\omega}_o > 0$ on both sides of the equation:

$$\int_{2\pi} T_{12} S_d T_{21} \cos\theta_o \mathrm{d}\vec{\omega}_o = \int_{2\pi} \eta^2 \frac{\mathrm{d}T_{21} L_d(\mathbf{x}_o, \vec{\omega}_{21})}{\mathrm{d}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)} \cos\theta_o \mathrm{d}\vec{\omega}_o \,.$$

Assuming no dependency on the outgoing direction, $S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o)$, and we have

$$T_{12} S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o) \int_{2\pi} T_{21} \cos\theta_o \mathrm{d}\vec{\omega}_o = \eta^2 \frac{\mathrm{d} \int_{2\pi} T_{21} L_d(\mathbf{x}_o, \vec{\omega}_{21}) \cos\theta_o \mathrm{d}\vec{\omega}_o}{\mathrm{d}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)}$$

$$T_{12} S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o) 4\pi \frac{C_\phi(\eta)}{\eta^2} = \frac{\mathrm{d}M_d(\mathbf{x}_o)}{\mathrm{d}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)}$$

$$T_{12} S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o) 4\pi C_\phi(1/\eta) = \frac{\mathrm{d}M_d(\mathbf{x}_o)}{\mathrm{d}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)} \,. \tag{8}$$

Let us calculate the diffuse radiant exitance first. We insert the diffusion approximation in place of L_d to find

$$M_d(\mathbf{x}_o) = \int_{2\pi} T_{21} L_d(\mathbf{x}_o, \vec{\omega}_{21}) \cos\theta_o \mathrm{d}\vec{\omega}_o = \int_{2\pi} T_{21} \left(\frac{\phi(\mathbf{x}_o)}{4\pi} - \frac{3}{4\pi} D \, \vec{\omega}_{21} \cdot \nabla\phi(\mathbf{x}_o) \right) \cos\theta_o \mathrm{d}\vec{\omega}_o$$

$$= \underbrace{\frac{\phi(\mathbf{x}_o)}{4\pi} \int_{2\pi} T_{21} \cos\theta_o \mathrm{d}\vec{\omega}_o}_{I_\phi} - \underbrace{\frac{3}{4\pi} D \int_{2\pi} \vec{\omega}_{21} \cdot \nabla\phi(\mathbf{x}_o) \cos\theta_o \mathrm{d}\vec{\omega}_o}_{I_\mathbf{E}} \,.$$

And, in a straightforward way,

$$I_\phi = C_\phi(\eta) \phi(\mathbf{x}_o) \,.$$

The second part is more complicated. Without loss of generality, we rotate the reference coordinate system so that $\vec{\omega}_o = (\cos\phi \sin\theta, \sin\phi \sin\theta, \cos\theta)$ and $\vec{n}_o = (0, 0, 1)$. Given this reference system, the refracted vector becomes:

$$\vec{\omega}_{21} = \frac{(\cos\phi \sin\theta, \sin\phi \sin\theta, \cos\theta)}{\eta} - \left(\frac{\cos\theta}{\eta} - \sqrt{1 - \frac{\sin^2\theta}{\eta^2}} \right) (0, 0, 1)$$

$$= \left(\frac{\cos\phi \sin\theta}{\eta}, \frac{\sin\phi \sin\theta}{\eta}, \sqrt{1 - \frac{\sin^2\theta}{\eta^2}} \right)$$

When inserted in the dot product in the expression for $I_\mathbf{E}$, we get a sum of three components. The first two terms of this sum are zero, since we can first integrate over ϕ:

$$\int_0^{2\pi} \cos\phi \mathrm{d}\phi = \int_0^{2\pi} \sin\phi \mathrm{d}\phi = 0 \,.$$

So we are left only with the cumbersome z term:

$$I_\mathbf{E} = \frac{3}{4\pi} D \frac{\partial \phi_z(\mathbf{x}_o)}{\partial z} \int_0^{2\pi} \int_0^{\frac{\pi}{2}} T_{21}(\eta, \theta_o) \sqrt{1 - \frac{\sin^2\theta_o}{\eta^2}} \cos\theta_o \sin\theta_o \mathrm{d}\theta_o \mathrm{d}\phi \,,$$

where we note that $\frac{\partial \phi_z}{\partial z} = \vec{n}_o \cdot \nabla\phi$. We assume $\eta > 1$, so that the argument of the square root is never negative. We now perform a substitution using the law of refraction, $\sin\theta_i = \eta \sin\theta_o$. We obtain the following identities:

$$\mathrm{d}\theta_o = \frac{\eta\cos\theta_i}{\cos\theta_o}\mathrm{d}\theta_i, \quad T_{21}(\eta,\theta_o) = T_{21}(1/\eta,\theta_i), \quad \sqrt{1-\frac{\sin^2\theta_o}{\eta^2}} = \cos\theta_i.$$

If we introduce a critical angle $\theta_c = \arcsin(1/\eta)$, we get

$$\begin{aligned} I_{\mathbf{E}} &= \frac{3}{4\pi} D\vec{n}_o \cdot \nabla\phi(\mathbf{x}_o) \int_0^{2\pi}\int_0^{\theta_c} T_{21}(1/\eta,\theta_i)\cos\theta_i\cos\theta_o\eta\sin\theta_i \frac{\eta\cos\theta_i}{\cos\theta_o}\mathrm{d}\theta_i\mathrm{d}\phi \\ &= D\vec{n}_o \cdot \nabla\phi(\mathbf{x}_o)\frac{3}{4\pi}\eta^2 \int_0^{2\pi}\int_0^{\theta_c} T_{21}(1/\eta,\theta_i)\cos^2\theta_i\sin\theta_i\mathrm{d}\theta_i\mathrm{d}\phi \\ &= D\vec{n}_o \cdot \nabla\phi(\mathbf{x}_o)\eta^2 C_{\mathbf{E}}(1/\eta) = C_{\mathbf{E}}(\eta) D\vec{n}_o \cdot \nabla\phi(\mathbf{x}_o)\,. \end{aligned}$$

We can then get our final expression for $M_d(\mathbf{x}_o)$

$$M_d(\mathbf{x}_o) = C_\phi(\eta)\phi(\mathbf{x}_o) - C_{\mathbf{E}}(\eta) D\vec{n}_o \cdot \nabla\phi(\mathbf{x}_o)\,.$$

Inserting into the expression (8), we derive the monopole BSSRDF:

$$S_d(\mathbf{x}_i,\vec{\omega}_i,\mathbf{x}_o) = \frac{1}{4\pi T_{12} C_\phi(1/\eta)} \frac{\mathrm{d}M_d(\mathbf{x}_o)}{\mathrm{d}\Phi_i(\mathbf{x}_i,\vec{\omega}_i)}$$

8.1 Diffuse monopole BSSRDF

Using the monopole solution (6) for a ray normally incident on an isotropic half-space:

$$\phi(\mathbf{x}) = \alpha' \frac{\Phi_i}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r}\,,$$

we can derive the gradient:

$$\nabla\phi(\mathbf{x}) = -\alpha' \frac{\Phi_i}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r^3}(1+\sigma_{tr}r)\mathbf{x}\,.$$

Thus, we can find the monopole BSSRDF for this configuration:

$$S_d(\mathbf{x}_i,\mathbf{x}_o) = \frac{1}{4C_\phi(1/\eta)}\frac{\alpha'}{4\pi^2}\frac{e^{-\sigma_{tr}r}}{r^3}\left[C_\phi(\eta)\frac{r^2}{D} + C_{\mathbf{E}}(\eta)(1+\sigma_{tr}r)(\mathbf{x}_o-\mathbf{x}_i)\cdot\vec{n}_o\right],$$

where we used $\Phi_i = T_{12}L_i(\mathbf{x}_i,\vec{\omega}_i)$. This is the monopole version of the better dipole from Eugene d'Eon [2]. To get a monopole version of the standard dipole [4, 6], we further approximate the above expression using $C_\phi(1/\eta) \approx 1/4$, $C_\phi(\eta) \approx 0$, and $C_{\mathbf{E}}(\eta) \approx 1$:

$$S_d(\mathbf{x}_i,\mathbf{x}_o) = \frac{\alpha'}{4\pi^2}\frac{e^{-\sigma_{tr}r}}{r^3}(1+\sigma_{tr}r)\mathbf{x}\cdot\vec{n}_o$$

This is the monopole version of the standard dipole.

8.2 Directional monopole BSSRDF

For the directional dipole, we have the following form:

$$\phi(\mathbf{x}) = \frac{\Phi}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r}\left(1 + 3D\frac{1+\sigma_{tr}r}{r}\cos\theta\right).$$

It is more convenient to do the gradient in spherical coordinates:

$$\nabla\phi(\mathbf{x}) = \frac{\partial}{\partial r}\phi(\mathbf{x})\vec{e}_r + \frac{1}{r}\frac{\partial}{\partial\theta}\phi(\mathbf{x})\vec{e}_\theta + \frac{1}{r\sin\theta}\frac{\partial}{\partial\phi}\phi(\mathbf{x})\vec{e}_\phi\,,$$

where

$$\frac{\partial}{\partial r}\phi(\mathbf{x}) = -\frac{\Phi}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r^2}\left(3D\frac{2(1+\sigma_{tr}r)+(\sigma_{tr}r)^2}{r}\cos\theta + (1+\sigma_{tr}r)\right)$$

and

$$\frac{\partial}{\partial\theta}\phi(\mathbf{x}) = -\frac{\Phi}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r^2}3D(1+\sigma_{tr}r)\sin\theta\,.$$

Finally, $\frac{\partial}{\partial\phi}\phi(\mathbf{x}) = 0$. Given our choice of basis for derivation, we can use the following identities:

$$\vec{e}_r = \frac{\mathbf{x}}{r}$$
$$-\vec{e}_\theta\sin\theta = \vec{\omega}_{12} - \vec{e}_r\cos\theta\,.$$

Inserting the identities, we get an expression for the gradient:

$$\nabla\phi(\mathbf{x}) = -\frac{\Phi}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r^2}\left(3D\frac{2(1+\sigma_{tr}r)+(\sigma_{tr}r)^2}{r}\cos\theta + (1+\sigma_{tr}r)\right)\vec{e}_r + \frac{1}{r}\frac{\Phi}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r^2}3D(1+\sigma_{tr}r)(-\vec{e}_\theta\sin\theta)$$

$$\nabla\phi(\mathbf{x}) = \frac{\Phi}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r^3}\left[\left(-3D\frac{2(1+\sigma_{tr}r)+(\sigma_{tr}r)^2}{r}\cos\theta - (1+\sigma_{tr}r)\right)r\vec{e}_r - 3D(1+\sigma_{tr}r)\cos\theta\vec{e}_r + 3D(1+\sigma_{tr}r)\vec{\omega}_{12}\right]$$

$$\nabla\phi(\mathbf{x}) = \frac{\Phi}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r^3}\left[-\left(3D\frac{3(1+\sigma_{tr}r)+(\sigma_{tr}r)^2}{r}\cos\theta + (1+\sigma_{tr}r)\right)\mathbf{x} + 3D(1+\sigma_{tr}r)\vec{\omega}_{12}\right].$$

We can now do the same as above, obtaining the directional monopole BSSRDF with $\mathbf{x} = \mathbf{x}_o - \mathbf{x}_i$ and $r = \|\mathbf{x}\|$:

$$S'_d(\mathbf{x}_i,\vec{\omega}_i,\mathbf{x}_o) = \frac{1}{4C_\phi(1/\eta)}\frac{1}{4\pi^2}\frac{e^{-\sigma_{tr}r}}{r^3}\left[C_\phi(\eta)\left(\frac{r^2}{D} + 3(1+\sigma_{tr}r)\mathbf{x}\cdot\vec{\omega}_{12}\right)\right.$$
$$\left.-C_{\mathbf{E}}(\eta)\left(3D(1+\sigma_{tr}r)\vec{\omega}_{12}\cdot\vec{n}_o - \left((1+\sigma_{tr}r) + 3D\frac{3(1+\sigma_{tr}r)+(\sigma_{tr}r)^2}{r^2}\mathbf{x}\cdot\vec{\omega}_{12}\right)\mathbf{x}\cdot\vec{n}_o\right)\right].$$

References

[1] Subrahmanyan Chandrasekhar. *Radiative Transfer*. Oxford, Clarendon Press, 1950. Unabridged and slightly revised version published by Dover Publications, Inc., in 1960.

[2] Eugene d'Eon. A better dipole. Publicly available technical report. http://www.eugenedeon.com/?project=a-better-dipole, 2012.

[3] W. G. Egan, T. Hilgeman, and J. Reichman. Determination of absorption and scattering coefficients for nonhomogeneous media. 2: Experiment. *Applied Optics*, 12(8):1816–1823, August 1973.

[4] Thomas J. Farrell, Michael S. Patterson, and Brian Wilson. A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties *in vivo*. *Medical Physics*, 19(4):879–888, July/August 1992.

[5] Jeppe Revall Frisvad, Toshiya Hachisuka, and Thomas Kim Kjeldsen. Directional dipole model for subsurface scattering. *ACM Transactions on Graphics*, 34(1):5:1–5:12, November 2014.

[6] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001*, pages 511–518. ACM, August 2001.

[7] J. H. Joseph, W. J. Wiscombe, and J. A. Weinman. The delta-Eddington approximation for radiative flux transfer. *Journal of Atmospheric Sciences*, 33:2452–2459, December 1976.

[8] S. Menon, Q. Su, and R. Grobe. Determination of g and μ using multiply scattered light in turbid media. *Physical Review Letters*, 94:153904, April 2005.

[9] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. Technical report, National Bureau of Standards (US), October 1977.

[10] M. S. Patterson, E. Schwartz, and B. C. Wilson. Quantitative reflectance spectrophotometry for the non-invasive measurement of photosensitizer concentration in tissue during photodynamic therapy. In *Proceedings of SPIE*, volume 1065 of *Photodynamic Therapy: Mechanisms*, pages 115–122, June 1989.

Appendix VII

Point cloud method for rendering BSSRDFs

Point cloud method for rendering BSSRDFs

Alessandro Dal Corso
Technical University of Denmark

Jeppe Revall Frisvad
Technical University of Denmark

March 2018

This is a short note on rendering of a triangle mesh onto which we apply a BSSRDF model. It is a way to implement the technique referred to by Frisvad et al. [1] as direct Monte Carlo integration. The method works for arbitrary triangle meshes, and it is unbiased. The configuration of the method is illustrated in Figure 1.

We first define some quantities relative to our mesh. We consider a triangle mesh a set $M = \{T_\Delta, \Delta \in [0, K-1]\}$ composed of K triangles. Each triangle T_Δ is composed of three vertices:

$$T_\Delta = \{\mathbf{v}_0^\Delta, \mathbf{v}_1^\Delta, \mathbf{v}_2^\Delta\}\,.$$

From these quantities, it straightforward to define two derived quantities, the normal $\vec{n}_\Delta$ and the area A_Δ of the triangle.

$$\vec{n}_\Delta = \frac{(\mathbf{v}_1^\Delta - \mathbf{v}_0^\Delta) \times (\mathbf{v}_2^\Delta - \mathbf{v}_0^\Delta)}{\|(\mathbf{v}_1^\Delta - \mathbf{v}_0^\Delta) \times (\mathbf{v}_2^\Delta - \mathbf{v}_0^\Delta)\|}$$

$$A_\Delta = \frac{1}{2}\|(\mathbf{v}_1^\Delta - \mathbf{v}_0^\Delta) \times (\mathbf{v}_2^\Delta - \mathbf{v}_0^\Delta)\|\,.$$

A triangle in a triangle mesh may have a different normal for each vertex. In this case, the normal $\vec{n}$ of a particular point $\mathbf{x}$ of the triangle is given by linear interpolation based on the barycentric coordinates of $\mathbf{x}$ in T_Δ. With our triangles defined, we can start describing our solution.

Theoretically, we solve the equation of reflected radiance for BSSRDFs:

$$L_r(\mathbf{x}_o, \vec{\omega}_o) = \int_A \int_\Omega S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_i, \vec{\omega}_i)(\vec{n}_i \cdot \vec{\omega}_i) d\omega_i dA\,,$$

where L_i and L_r are incident and reflected radiance, S is the BSSRDF, A is the surface area of the object represented by the triangle mesh, and Ω is the hemisphere around the surface normal $\vec{n}_i$ at the surface position $\mathbf{x}_i$. We now solve the integral using Monte Carlo integration with importance sampling. Taking M surface position samples for the area integral and N direction samples for the direction integral, we have

$$\begin{aligned}\hat{L}_r(\mathbf{x}_o, \vec{\omega}_o) &\approx \frac{1}{NM}\sum_{p=1}^{M}\sum_{q=1}^{N} \frac{S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})(\vec{n}_{i,p} \cdot \vec{\omega}_{i,q})}{\mathrm{pdf}(\mathbf{x}_{i,p})\mathrm{pdf}(\vec{\omega}_{i,q})} \\ &= \frac{1}{NM}\sum_{p=1}^{M}\sum_{q=1}^{N} S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o)\Phi_{i,p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})\,. \qquad (1)\end{aligned}$$

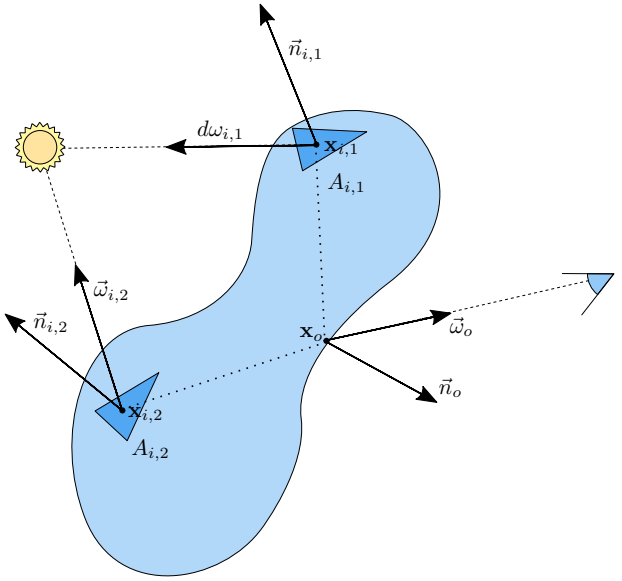


Figure 1: Diagram illustrating the configuration. Two points $\mathbf{x}_{i,p}$, $p = 1, 2$, are selected on the surface, and the irradiance stored in a buffer. The contribution is then gathered from the various points of emergence $\mathbf{x}_o$ in the outgoing direction $\vec{\omega}_o$ towards the eye.

The algorithm is composed of two phases. In the first phase, we generate the NM samples on the surface of the model, storing the incoming flux $\Phi_{i,p,q}$ for each position $\mathbf{x}_{i,p}$ and direction $\vec{\omega}_{i,q}$. In the second phase, we render the final image evaluating the contribution at each pixel.

In phase one, we produce NM samples. Sample p, q is composed of a position $\mathbf{x}_{i,p}$, a direction $\vec{\omega}_{i,q}$, and a flux $\Phi_{i,p,q}$. First, we randomly sample a triangle T_Δ. This is done by uniformly sampling a triangle index Δ, so given the continuous uniform random variable $\xi \in [0, 1)$, we have $\Delta = \lfloor \xi K \rfloor$. The triangle index could be importance sampled using A_Δ / A as the probability of sampling index Δ, but this requires a binary search, which turned out to be less efficient on a GPU. To get $\mathbf{x}_{i,p}$, we uniformly sample a point in the triangle T_Δ using barycentric coordinates:

$$\mathbf{x}_{i,p} = (1 - \sqrt{\xi_0})\mathbf{v}_0^j + (1 - \xi_1)\sqrt{\xi_0}\mathbf{v}_1^j + \xi_1\sqrt{\xi_0}\mathbf{v}_2^j\,,$$

where $\xi_0, \xi_1 \in [0, 1)$ are again continuous uniform random variables.

Now, the sampling direction $\vec{\omega}_{i,q}$ and the flux $\Phi_{i,p,q}$ need to be evaluated. We leave the choice of the sampling distribution for $\vec{\omega}_{i,q}$ to implementation, since it is not important for the method. From sampling the light source, we obtain $\vec{\omega}_{i,q}$ and an irradiance $E_{p,q}$:

$$E_{p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) = \frac{L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})(\vec{n}_{i,p} \cdot \vec{\omega}_{i,q})}{\mathrm{pdf}(\vec{\omega}_{i,q})}$$

We need the pdf for the sampling of the point of incidence $\text{pdf}(\mathbf{x}_{i,p})$. Since we uniformly sampled a triangle index and then uniformly a point within the triangle, the pdf is

$$\text{pdf}(\mathbf{x}_{i,p}) = \frac{1}{KA_\Delta}$$

Putting it all together:

$$\Phi_{i,p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) = \frac{E_{p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})}{\text{pdf}(\mathbf{x}_{i,p})} = KA_\Delta E_{p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})\,.$$

Once we have stored this quantity in the sample, we are ready to proceed to the second and final phase. For each pixel, we ray trace a camera ray, obtaining a point $\mathbf{x}_o$ and a direction towards the camera $\vec{\omega}_o$. We consider the case of a participating medium with scattering properties σ_s, σ_a, g, relative index of refraction η, and a perfectly smooth surface so that the Fresnel equations for reflection describe the scattering at the surface.

In our path tracing implementation, we perform a Russian roulette using the Fresnel reflectance R as the probability of reflection when choosing reflection or refraction. In the case of reflection, we continue tracing the reflected ray. In the case of refraction, we calculate the final radiance as:

$$\hat{L}_o(\mathbf{x}_o, \vec{\omega}_o) = L_e(\mathbf{x}_o, \vec{\omega}_o) + L_t(\mathbf{x}_o, \vec{\omega}_o) + \hat{L}_r(\mathbf{x}_o, \vec{\omega}_o)\,,$$

where L_e is the emitted radiance from the medium, L_t is the direct transmission (or reduced intensity) term, and $\hat{L}_r$ comes from the Monte Carlo estimator (1).

To efficiently render the model and save expensive BSSRDF evaluations, we do a last optimization. We probabilistically include or reject samples at position $\mathbf{x}_{i,p}$ with $\exp(-\sigma_{tr}\|\mathbf{x}_o - \mathbf{x}_k\|)$ being the probability of inclusion, where $\sigma_{tr} = \sqrt{3\sigma_a(\sigma_a + (1-g)\sigma_s)}$ is the effective transport coefficient. So, only samples that are very close to the exit point $\mathbf{x}_o$ will actually be evaluated. In a formula,

$$\hat{L}_r(\mathbf{x}_o, \vec{\omega}_o) \approx \frac{1}{NM} \sum_{p=1}^{M} \sum_{q=1}^{N} S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o) \Phi_{i,p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) V(\xi, e^{-\sigma_{tr}\|\mathbf{x}_o - \mathbf{x}_{i,p}\|})\; e^{\sigma_{tr}\|\mathbf{x}_o - \mathbf{x}_{i,p}\|}\,,$$

where:

$$V(\xi, d) = \begin{cases} 1 & \text{if } \xi < d \\ 0 & \text{otherwise}\,. \end{cases}$$

We always use $N = 1$, and we generate a new set of M samples for each frame in a progressive path tracing. In case of RGB rendering, we use the mean of the three color components in σ_{tr} when sampling for inclusion or rejection.

References

[1] Jeppe Revall Frisvad, Toshiya Hachisuka, and Thomas Kim Kjeldsen. Directional dipole model for subsurface scattering. *ACM Transactions on Graphics*, 34(1):5:1–5:12, November 2014.

Appendix VIII

On BSSRDF estimation

On BSSRDF estimation

Alessandro Dal Corso
Technical University of Denmark

Jeppe Revall Frisvad
Technical University of Denmark

January 2018

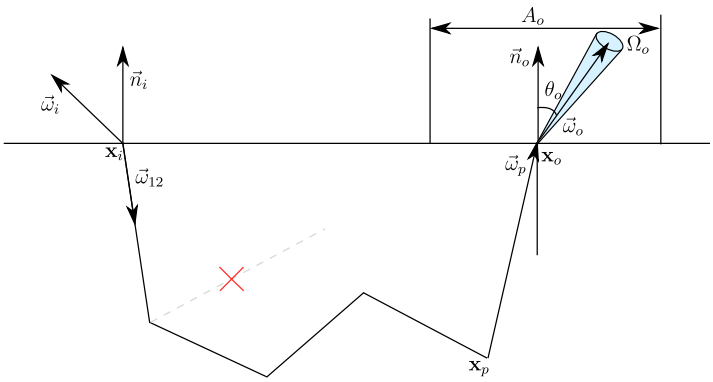


Figure 1: Configuration for random walk procedure.

The configuration of the bidirectional scattering-surface reflectance distribution function (BSSRDF) is illustrated in Figure 1. Our aim is to estimate the BSSRDF for a specific spatial area bin with area A_o and for a specific angular bin with solid angle Ω_o. In a measurement environment [4], we have the following incident flux:

$$\Phi_i(A_i, \Omega_i) = \int_{A_i} \int_{\Omega_i} L_i(\mathbf{x}'_i, \vec{\omega}'_i)(\vec{\omega}'_i \cdot \vec{n}_i)\, d\omega'_i\, dA'_i\,,$$

where L_i is radiance incident somewhere on the surface area A_i from the solid angle Ω_i. As illustrated in Figure 1, $\vec{n}_i$ is the surface normal at a point of incidence $\mathbf{x}'_i$, while $\vec{\omega}'_i$ is a direction of incidence. The corresponding responsivity-weighted emergent flux is:

$$\Phi_o(A_o, \Omega_o) = \int_{A_o} \int_{\Omega_o} \int_{A_i} \int_{\Omega_i} L_i(\mathbf{x}'_i, \vec{\omega}'_i) S(\mathbf{x}'_i, \vec{\omega}'_i, \mathbf{x}'_o, \vec{\omega}'_o) R(\mathbf{x}'_o, \vec{\omega}'_o)(\vec{\omega}'_i \cdot \vec{n}_i)\, d\omega'_i\, dA_i\, (\vec{\omega}'_o \cdot \vec{n}_o)\, d\omega'_o\, dA'_o\,,$$

where R is the relative responsivity of the instrument used for a measurement and S is the BSSRDF. Note that we choose our notation so that $\Phi_o(A_o, \Omega_o)$ rep-

resents the emergent flux arriving in the bin of interest. In an idealized computational setting, we ignore the responsivity of the sensor and set $R(\mathbf{x}'_o, \vec{\omega}'_o) = 1$. If we choose a unit response function and unit incident radiance $L_i(\mathbf{x}_i, \vec{\omega}_i) = \delta(\mathbf{x}_i - \mathbf{x}'_i)\delta(\vec{\omega}_i - \vec{\omega}'_i)$, the ratio of emergent to incident flux is

$$\rho(A_o, \Omega_o) = \frac{\Phi_o(A_o, \Omega_o)}{\Phi_i(A_i, \Omega_i)} = \int_{A_o} \int_{\Omega_o} S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}'_o, \vec{\omega}'_o)(\vec{\omega}'_o \cdot \vec{n}_o)\, d\omega'_o\, dA'_o\,.$$

Once we have obtained ρ, we can rearrange the above equation to obtain the BSSRDF of a bin with $\mathbf{x}_o \in A_o$ and $\vec{\omega}_o \in \Omega_o$:

$$S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = \frac{dL_o(\mathbf{x}_o, \vec{\omega}_o)}{d\Phi_i(\mathbf{x}_i, \vec{\omega}_i)} \approx \frac{\Phi_o(A_o, \Omega_o)}{\Phi_i(A_i, \Omega_i) A_o \Omega_o (\vec{\Omega}_o \cdot \vec{n}_o)} = \frac{\rho(A_o, \Omega_o)}{A_o \Omega_o (\vec{\Omega}_o \cdot \vec{n}_o)}\,, \tag{1}$$

where $\vec{\Omega}_o$ is the direction around which the bin solid angle is centered. This means that we can represent the light scattering between two points, that is, the BSSRDF, as the ratio of emergent to incoming flux divided by the projected area and the solid angle.

The BSSRDF of a medium can be defined as an operator that includes all radiance resulting from the scattering events in the medium [2]. This radiance is described locally by the radiative transfer equation [2]. Thus, we can evaluate the scattering process given by the BSSRDF using the radiative transfer equation. So, the reflectance quantity ρ becomes the ratio of flux carried by a flux packet that arrives in a bin after going through a random walk. This is the procedure used by Wang et al. [5].

The random walk procedure works as follows:

1. Create a new photon with flux $\Phi_t = 1/N$, where N is the number of photons. Start assigning $\mathbf{x}_p = \mathbf{x}_i$ and $\vec{\omega}_p = \vec{\omega}_{12}$, the refracted incoming direction.

2. Sample a new distance to the next scattering event s. We assume a homogenous medium, so we use an exponential distribution with decay σ_t, that can be easily importance sampled:

$$s = \frac{-\log(1-\xi)}{\sigma_t}\,, \quad \text{with } \xi \in [0, 1).$$

3. Check if the next scattering event $\mathbf{x}'_p = \mathbf{x}_p + s\vec{\omega}_p$ is within the medium.

4. If the next scattering event is within the medium:

   (a) Terminate the path with probability $1 - \alpha$. If terminated, start from step 1.

   (b) If not terminated, sample a new direction $\vec{\omega}_p$ according to the phase function.

   (c) Continue from step 2.

5. If the next scattering event is outside of the medium:

6. Calculate the reflection Fresnel coefficient R_{21} and update $\mathbf{x}_p$ to be the intersection point on the surface of the medium.

(a) With probability R_{21} reflect the path around $-\vec{n}_o$, and continue from step 2.

(b) With probability $1 - R_{21}$ refract the path outside, storing Φ_t in the corresponding bin. Start from step 1.

Note that all the sampling is chosen to cancel out the terms that would change the stored flux. So, the flux within the medium should change for each scattering event as:

$$\Phi_t' = \Phi_t \sigma_s p(\vec{\omega}_p \cdot \vec{\omega}_p) e^{-\sigma_t s} .$$

Although, dividing by the pdfs, $p(\vec{\omega}_p \cdot \vec{\omega}_p')$ cancels out if we sample $\vec{\omega}_p$ according to the correct phase function, $e^{-\sigma_t s}$ cancels with the pdf of the distance sampling, leaving $\sigma_s/\sigma_t = \alpha$. This last factor is divided out by the path continuation probability, leading to $\Phi_t' = \Phi_t$ as expected. If the flux goes outside the medium, the Fresnel coefficients cancel with the probabilities of reflection or transmission.

1 Variance reduction through connections

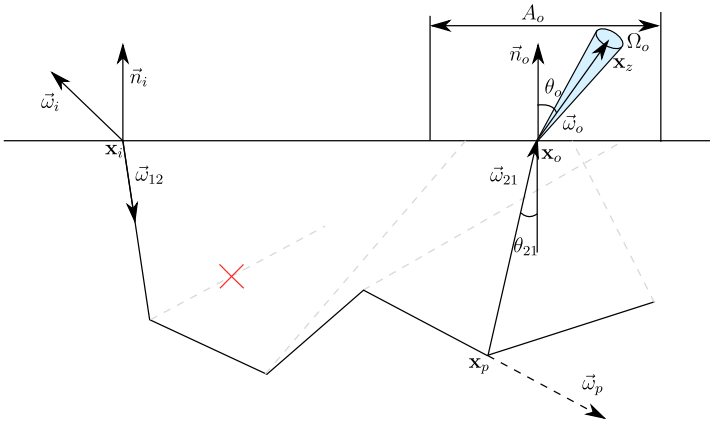


Figure 2: Configuration for the connection event described in the text.

The procedure from the previous section provides an unbiased estimate of the BSSRDF. However, this process requires a huge number of photons to converge to an acceptable solution. To reduce convergence times, we connect each scattering event (except the first, if we measure multiple scattering only), to a point $\mathbf{x}_o$ on the surface. We then use the three point form of the local scattering equation [3] to calculate the resulting measurement:

$$L(\mathbf{x}_o, \vec{\omega}_o) = L_e(\mathbf{x}_o, \vec{\omega}_o) + L(\mathbf{x}_p \to \mathbf{x}_o) f(\mathbf{x}_p \to \mathbf{x}_o \to \mathbf{x}_z) G(\mathbf{x}_p \leftrightarrow \mathbf{x}_o) V(\mathbf{x}_p \leftrightarrow \mathbf{x}_o) .$$

We assume a non emissive body ($L_e(\mathbf{x}_o, \vec{\omega}_o) = 0$). In our case, we have a point in the volume generating a scattering event ($\mathbf{x}_p$), a point in a bin on the surface ($\mathbf{x}_o$), and a point specifying a direction into the bin ($\mathbf{x}_z$), see Figure 2. So, we have the incoming radiance:

$$L(\mathbf{x}_p \to \mathbf{x}_o) = \alpha \Phi_t(\mathbf{x}_p) p(\vec{\omega}_p \cdot \vec{\omega}_{21}, g) ,$$

the three point scattering function corresponding to the bidirectional transmittance distribution function (BTDF) of a perfectly smooth surface [1]:

$$\begin{aligned} f(\mathbf{x}_p \to \mathbf{x}_o \to \mathbf{x}_z) &= T_{21}(\eta, \vec{\omega}_o)\frac{\delta(\vec{\omega}_o - \vec{\omega}_t)}{|\vec{\omega}_{21} \cdot \vec{n}_o|}\,, \\ \vec{\omega}_t &= \eta^{-1}(\vec{\omega}_{21} - (\vec{\omega}_{21} \cdot \vec{n}_o)\vec{n}_o) + \vec{n}_o\sqrt{1 - \eta^{-2}(1 - (\vec{\omega}_{21} \cdot \vec{n}_o)^2)}\,, \end{aligned} \tag{2}$$

the geometry term:

$$G(\mathbf{x}_p \leftrightarrow \mathbf{x}_o) = \frac{|\vec{\omega}_{21} \cdot \vec{n}_o|}{\|\mathbf{x}_o - \mathbf{x}_p\|^2}\,,$$

and the generic visibility term:

$$V(\mathbf{x}_p \leftrightarrow \mathbf{x}_o) = V'(\mathbf{x}_p \leftrightarrow \mathbf{x}_o)\exp\left(-\int_0^{\|\mathbf{x}_o - \mathbf{x}\|} \sigma_t\left(\mathbf{x}_p + t\vec{\omega}_{21}\right) dt\right).$$

Here, $T_{21} = 1 - R_{21}$ is the Fresnel transmittance at the point of emergence, and one should note that $T_{21}(\eta^{-1}, -\vec{\omega}_{21}) = T_{21}(\eta, \vec{\omega}_o)$. In addition, we disregard (de)compression of solid angle in the BTDF, as we consider light incident and emergent in the same medium only, so the compression cancels out upon emergence.

In our configuration of a semi infinite plane, the binary visibility function $V'(\mathbf{x}_p \leftrightarrow \mathbf{x}_o)$ is always one (i.e., it is always possible to connect to the emergent point). Since we have a homogenous medium, we can rewrite V as the beam transmittance:

$$V(\mathbf{x}_p \leftrightarrow \mathbf{x}_o) = \exp(-\sigma_t\|\mathbf{x}_o - \mathbf{x}_p\|)\,.$$

And the radiance of the pth scattering event of the kth photon becomes

$$L_{k,p}(\mathbf{x}_o, \vec{\omega}_o) = \alpha\Phi_t(\mathbf{x}_p)p(\vec{\omega}_p \cdot \vec{\omega}_{21}, g)T_{21}(\eta, \vec{\omega}_o)\frac{\delta(\vec{\omega}_o - \vec{\omega}_t)}{\|\mathbf{x}_o - \mathbf{x}_p\|^2}\exp(-\sigma_t\|\mathbf{x}_o - \mathbf{x}_p\|)\,.$$

To get the collective contribution of a photon path, we need to sum all the elements of the random walk:

$$L_k(\mathbf{x}_o, \vec{\omega}_o) = \sum_p L_{k,p}(\mathbf{x}_o, \vec{\omega}_o)\,.$$

And to get the final emergent flux, we need to solve the measurement equation for each bin. This is integration of incident flux (given by the three point form of the local scattering equation) across the cosine-weighted area and the solid angle of the bin:

$$\Phi_o(A_o, \Omega_o) = \int_{A_o}\int_{\Omega_o} L(\mathbf{x}'_o, \vec{\omega}'_o)(\vec{\omega}'_o \cdot \vec{n}_o)\, d\omega'_o\, dA'_o\,.$$

Now, since the three point scattering function (2) is a delta function with respect to direction, the integration over bin solid angle cancels with this delta. As for the area integral, we solve it using Monte Carlo integration with $\mathbf{x}_o$ sampled in polar coordinates:

$$\Phi_o(A_o, \Omega_o) = \frac{1}{M}\sum_{q=1}^{M}\sum_{k=1}^{N}\frac{L_k(\mathbf{x}_{o,q}, \vec{\omega}_t)w(\Omega_o, \vec{\omega}_t)\|\mathbf{x}_{o,q} - \mathbf{x}_i\|}{\mathrm{pdf}(\mathbf{x}_{o,q})}\,,$$

where $w(\Omega_o, \vec{\omega}_t)$ is 1 if $\vec{\omega}_t \in \Omega_o$ and 0 otherwise. Using just one area sample per bin per progressive update ($M = 1$, $\mathbf{x}_{o,q} = \mathbf{x}_{o,1} = \mathbf{x}'_o$) and recalling that $\Phi_t = 1/N$, the estimator becomes

$$\Phi_o(A_o, \Omega_o) = \frac{\alpha}{N} \sum_{k=1}^{N} \sum_{p} p(\vec{\omega}_p \cdot \vec{\omega}_{21}, g) T_{21}(\eta, \vec{\omega}_t) \frac{\vec{\omega}_t \cdot \vec{n}_o}{\|\mathbf{x}'_o - \mathbf{x}_p\|^2} e^{-\sigma_t \|\mathbf{x}'_o - \mathbf{x}_p\|} \frac{\|\mathbf{x}'_o - \mathbf{x}_i\|}{\mathrm{pdf}(\mathbf{x}'_o)} w(\Omega_o, \vec{\omega}_t)\,,$$

where we sample $\mathbf{x}'_o$ uniformly in polar coordinates. If we think of our spatial bin A_o as a circular sector delimited by $r_{\min}$ and $r_{\max}$ in radius, and $\theta_{s,\min}$ and $\theta_{s,\max}$. We have $\mathrm{pdf}(\mathbf{x}'_o) = 1/(\Delta r \Delta \theta_s)$, where $\Delta r = r_{\max} - r_{\min}$ and $\Delta\theta_s = \theta_{s,\max} - \theta_{s,\min}$. Since we use a circular sector, we obtain:

$$A_o = \int_{A_o} dA = \int_{r_{\min}}^{r_{\max}} \int_{\theta_{s,\min}}^{\theta_{s,\max}} r dr d\theta_s = \frac{r_{\max}^2 - r_{\min}^2}{2} \Delta\theta_s = \frac{r_{\min} + r_{\max}}{2} \Delta r \Delta \theta_s\,.$$

Also, since we are using a uniform sampling of the hemisphere, we obtain a constant solid angle for all directional bins:

$$\Omega_o = \frac{2\pi}{N_{bins}}\,.$$

Plugging everything into the original BSSRDF equation (1) and simplifying, we obtain the final BSSRDF estimate:

$$\begin{aligned} S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = & \frac{\alpha}{N} \sum_{k=1}^{N} \sum_{p} p(\vec{\omega}_p \cdot \vec{\omega}_{21}, g) \frac{T_{21}(\eta, \vec{\omega}_t)}{\vec{\Omega}_o \cdot \vec{n}_o} \frac{\vec{\omega}_t \cdot \vec{n}_o}{\|\mathbf{x}'_o - \mathbf{x}_p\|^2} e^{-\sigma_t \|\mathbf{x}'_o - \mathbf{x}_p\|} \\ & \|\mathbf{x}'_o - \mathbf{x}_i\| \frac{2}{r_{\min} + r_{\max}} \frac{N_{bins}}{2\pi} w(\Omega_o, \vec{\omega}_t) \end{aligned}$$

for $\mathbf{x}_o \in A_o$ and $\vec{\omega}_o \in \Omega_o$.

References

[1] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann/Elsevier, third edition, 2017.

[2] R. W. Preisendorfer. *Radiative Transfer on Discrete Spaces*. Pergamon Press, Oxford, 1965.

[3] Matthias Raab, Daniel Seibert, and Alexander Keller. Unbiased global illumination with participating media. In Alexander Keller, Stefan Heinrich, and Harald Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 591–605, Berlin, Heidelberg, 2008. Springer.

[4] William H. Venable, Jr. and Jack J. Hsia. Optical Radiation Measurements: Describing Spectrophotometric Measurements. Technical Report NBS-TN-594-9, November 1974.

[5] Lihong Wang, Steven L. Jacques, and Liqiong Zheng. MCML — Monte Carlo modeling of light transport in multi-layered tissues. *Computer Methods and Programs in Biomedicine*, 47(2):131 – 146, 1995.

Appendix IX

Camera space BSSRDF importance sampling

Camera space BSSRDF importance sampling

Alessandro Dal Corso
Technical University of Denmark

September 2017

This note explain the importance sampling technique in camera space from Mertens et al. [1].

1 Planar surfaces

We start from the classical BSSRDF form of the rendering equation:

$$L_o(\mathbf{x}_o, \vec{\omega}_o) = \int_A \int_{\Omega(\mathbf{x}_i)} L_i(\mathbf{x}_i, \vec{\omega}_i) S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o)(\vec{\omega}_i \cdot \vec{n}_i) d\omega_i dA$$

Where $\Omega(\mathbf{x}_i)$ is an hemisphere placed in $\mathbf{x}_i$ with normal $\vec{n}_i$. Our first step is to convert the integration into polar coordinates. We assume for now that the integration happens on a plane. In polar coordinates, we have $dA = r dr d\theta$. The integral becomes:

$$L_o(\mathbf{x}_o, \vec{\omega}_o) = \int_0^{2\pi} \int_0^{+\infty} \int_{\Omega(\mathbf{x}_i)} L_i(\mathbf{x}_i, \vec{\omega}_i) S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o)(\vec{\omega}_i \cdot \vec{n}_i) d\omega_i r dr d\theta$$

Where $r = \|\mathbf{x}_o - \mathbf{x}_i\|$. Also, $\mathbf{x}_i = \mathbf{x}_o + r\cos\theta\ \vec{t}_o + r\sin\theta\ \vec{b}_o$, where $\vec{t}_o$ and $\vec{b}_o$ form an orthonormal basis with $\vec{n}_o$.

We can now perform Monte Carlo integration, that gives the estimator:

$$L_o^{N,M}(\mathbf{x}_o, \vec{\omega}_o) = \frac{1}{NM} \sum_{p=1}^{M} \sum_{q=1}^{N} \frac{L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o)(\vec{\omega}_{i,q} \cdot \vec{n}_{i,p}) r}{\text{pdf}(r, \theta)\text{pdf}(\omega_{i,q})}$$

Now, we can importance sample our disc coordinates (r, θ) using the joint pdf:

$$\text{pdf}(r, \theta) = \frac{1}{2\pi} \sigma_{tr} e^{-\sigma_{tr} r}$$

And we can sample the incoming light direction using a cosine weighted hemisphere:

$$\text{pdf}(\omega_{i,q}) = \frac{\vec{\omega}_{i,q} \cdot \vec{n}_{i,p}}{\pi}$$

So the integral can be finally approximated as:

$$L_o^{N,M}(\mathbf{x}_o, \vec{\omega}_o) = \frac{2\pi^2}{NM\sigma_{tr}} \sum_{p=1}^{M} \sum_{q=1}^{N} L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o) r e^{\sigma_{tr} r}$$

Note that the $re^{\sigma_{tr} r}$ cancels out some terms inside the BSSRDF.

2 Non planar surfaces

We now assume that the surface is no longer planar. So we want to make a change of variables so that our integral becomes tractable, i.e. we are able to perform it in the tangent plane. Let us call dA_{tan} an element of area in the tangent plane. The change of variables simply becomes:

$$L_o(\mathbf{x}_o, \vec{\omega}_o) = \int_{A_{tan}} \int_{\Omega(\mathbf{x}_i)} L_i(\mathbf{x}_i, \vec{\omega}_i) S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o)(\vec{\omega}_i \cdot \vec{n}_i) d\omega_i \left| \frac{dA}{dA_{tan}} \right| dA_{tan}$$

Where dA is the corresponding area element on the real surface, continuing a camera ray. We call the camera ray $\vec{d} = \frac{\mathbf{x}_i - \mathbf{e}}{\|\mathbf{x}_i - \mathbf{e}\|}$, where $\mathbf{e}$ is the camera position.

We only need to estimate the Jacobian of the change of variables $\left| \frac{dA}{dA_{tan}} \right|$. We observe that, since we are using a pinhole camera, the solid angles subtended by the projected area elements must be equal. We calculate them and put them equal, that gives:

$$\frac{dA(\vec{n}_i \cdot -\vec{d})}{d^2} = \frac{dA_{tan}(\vec{n}_o \cdot -\vec{d})}{d_{tan}^2}$$

Where $d = \|\mathbf{x}_i - \mathbf{e}\|$ and $d_{tan} = \|\mathbf{x}_{i,tan} - \mathbf{e}\|$ are the distances with the real point and the point in tangent space, respectively. The Jacobian then becomes:

$$\left| \frac{dA}{dA_{tan}} \right| = \frac{(\vec{n}_o \cdot -\vec{d})}{(\vec{n}_i \cdot -\vec{d})} \frac{d^2}{d_{tan}^2}$$

Note that for a plane, $\vec{n}_i = \vec{n}_o$ and $d = d_{tan}$, giving a unitary Jacobian as expected.

After the change of variables, we can now proceed as in the previous section, and obtaining the final estimator:

$$L_o^{N,M}(\mathbf{x}_o, \vec{\omega}_o) = \frac{2\pi^2}{NM\sigma_{tr}} \sum_{p=1}^{M} \sum_{q=1}^{N} L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o) \frac{(\vec{n}_o \cdot -\vec{d})}{(\vec{n}_i \cdot -\vec{d})} \frac{d^2}{d_{tan}^2} r e^{\sigma_{tr} r}$$

References

[1] Tom Mertens, Jan Kautz, Philippe Bekaert, Hans-Peter Seidel, and Frank Van Reeth. Interactive rendering of translucent deformable objects. In *Proceedings of Eurographics Symposium on Rendering (EGSR 2003)*, pages 130–140, June 2003.

Bibliography

Henrik Aanæs, Knut Conradsen, Alessandro Dal Corso, Anders Bjorholm Dahl, A Del Bue, Mads Emil Brix Doest, Jeppe Revall Frisvad, Sebastian Hoppe Nesgaard Jensen, Jannik Boll Nielsen, Jonathan Dyssel Stets, et al. Our 3D vision data-sets in the making. In *CVPR Workshop: The Future of Datasets in Vision 2015*, 2015.

Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), pages 37–45, New York, NY, USA, 1968. ACM. doi: 10.1145/1468075.1468082. URL http://doi.acm.org/10.1145/1468075.1468082.

James Arvo and David Kirk. Particle transport and image synthesis. *Computer Graphics (Proceedings of SIGGRAPH '90*, 24(4):63–66, August 1990.

James F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. *Computer Graphics (Proceedings of ACM SIGGRAPH 82)*, 16(3): 21–29, July 1982.

Jesper Børlum, Brian Bunch Christensen, Thomas Kim Kjeldsen, Peter Trier Mikkelsen, Karsten Østergaard Noe, Jens Rimestad, and Jesper Mosegaard. SSLPV: Subsurface light propagation volumes. In *Proceedings of ACM SIGGRAPH Symposium on High Performance Graphics (HPG '11)*, pages 7–14, August 2011.

M. Bunnel. Dynamic ambient occlusion and indirect lighting. In M. Pharr, editor, *GPU Gems 2*, pages 223–233. Addison-Wesley, 2005.

N. Bus, N. H. Mustafa, and V. Biri. Global illumination using well separated pair decomposition. *Computer Graphics Forum*, 34(8):88–103, 2015. doi:

10.1111/cgf.12610. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12610.

K.M. Case and P.F. Zweifel. *Linear transport theory*. Addison-Wesley series in nuclear engineering. Addison-Wesley Pub. Co., 1967. URL https://books.google.dk/books?id=JZE8AAAAIAAJ.

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graph.*, 36(4):98:1–98:12, July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073601. URL http://doi.acm.org/10.1145/3072959.3073601.

Subrahmanyan Chandrasekhar. *Radiative Transfer*. Oxford, Clarendon Press, 1950. Unabridged and slightly revised version published by Dover in 1960.

Subrahmanyan Chandrasekhar. On the diffuse reflection of a pencil of radiation by a plane-parallel atmosphere. *Proceedings of the National Academy of Sciences of the United States of America*, 44(9):933–940, September 1958.

Per H. Christensen. Point-based approximate color bleeding. Technical Report Pixar Technical Memo, Pixar, 2008.

James H. Clark. Hierarchical geometric models for visible surface algorithms. *Commun. ACM*, 19(10):547–554, October 1976. ISSN 0001-0782. doi: 10.1145/360349.360354. URL http://doi.acm.org/10.1145/360349.360354.

Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive indirect illumination using voxel cone tracing: A preview. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 207–207, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0565-5. doi: 10.1145/1944745.1944787. URL http://doi.acm.org/10.1145/1944745.1944787.

Cyril Crassin, Morgan McGuire, Kayvon Fatahalian, and Aaron Lefohn. Aggregate G-buffer anti-aliasing. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, i3D '15, pages 109–119, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3392-4. doi: 10.1145/2699276.2699285. URL http://doi.acm.org.proxy.findit.dtu.dk/10.1145/2699276.2699285.

Carsten Dachsbacher and Marc Stamminger. Reflective shadow maps. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D '05, pages 203–231, New York, NY, USA, 2005. ACM. ISBN 1-59593-013-2. doi: 10.1145/1053427.1053460. URL http://doi.acm.org/10.1145/1053427.1053460.

Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. Scalable realistic rendering with many-light methods. *Comput. Graph. Forum*, 33(1):88–104, February 2014. ISSN 0167-7055. doi: 10.1111/cgf.12256. URL http://dx.doi.org/10.1111/cgf.12256.

A. Dal Corso, M. Olsen, K. H. Steenstrup, J. Wilm, S. Jensen, R. R. Paulsen, E. Eiríksson, J. Nielsen, J. R. Frisvad, G. Einarsson, and H. M. Kjer. Virtualtable: A projection augmented reality game. In *SIGGRAPH Asia 2015 Posters*, SA '15, pages 40:1–40:1, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3926-1. doi: 10.1145/2820926.2820950. URL http://doi.acm.org/10.1145/2820926.2820950.

Alessandro Dal Corso, Jeppe Revall Frisvad, Thomas Kim Kjeldsen, and Jakob Andreas Bærentzen. Interactive appearance prediction for cloudy beverages. In Reinhard Klein and Holly Rushmeier, editors, *Workshop on Material Appearance Modeling*. The Eurographics Association, 2016. ISBN 978-3-03868-007-9. doi: 10.2312/mam.20161247.

Alessandro Dal Corso, Jeppe Revall Frisvad, Jesper Mosegaard, and J. Andreas Bærentzen. Interactive directional subsurface scattering and transport of emergent light. *The Visual Computer*, 33(3):371–383, Mar 2017a. ISSN 1432-2315. doi: 10.1007/s00371-016-1207-2. URL https://doi.org/10.1007/s00371-016-1207-2.

Alessandro Dal Corso, Marco Salvi, Craig Kolb, Jeppe Revall Frisvad, Aaron Lefohn, and David Luebke. Interactive stable ray tracing. In *Proceedings of High Performance Graphics*, HPG '17, pages 1:1–1:10, New York, NY, USA, 2017b. ACM. ISBN 978-1-4503-5101-0. doi: 10.1145/3105762.3105769. URL http://doi.acm.org/10.1145/3105762.3105769.

Alessandro Dal Corso, Jonathan Dyssel Stets, Andrea Luongo, Jannik Boll Nielsen, Jeppe Revall Frisvad, and Henrik Aanæs. Virtual reality inspection and painting with measured BRDFs. In *SIGGRAPH Asia 2017 VR Showcase*, SA '17, pages 15:1–15:2, New York, NY, USA, 2017c. ACM. ISBN 978-1-4503-5408-0. doi: 10.1145/3139468.3139472. URL http://doi.acm.org/10.1145/3139468.3139472.

Tomáš Davidovič, Jaroslav Křivánek, Miloš Hašan, and Philipp Slusallek. Progressive light transport simulation on the GPU: Survey and improvements. *ACM Trans. Graph.*, 33(3):29:1–29:19, June 2014. ISSN 0730-0301. doi: 10.1145/2602144. URL http://doi.acm.org.proxy.findit.dtu.dk/10.1145/2602144.

Eugene d'Eon. A better dipole. Publicly available technical report. http://www.eugenedeon.com/?project=a-better-dipole, 2012.

Eugene d'Eon and Geoffrey Irving. A quantized-diffusion model for rendering translucent materials. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011)*, 30(4):56:1–56:13, July 2011.

Zhao Dong, Thorsten Grosch, Tobias Ritschel, and Hans peter Seidel. H.-p.: Real-time indirect illumination with clustered visibility. In *In Proc. Vision Modeling and Visualization*, 2009.

Craig Donner, Jason Lawrence, Ravi Ramamoorthi, Toshiya Hachisuka, Henrik Wann Jensen, and Shree Nayar. An empirical BSSRDF model. *ACM Trans. Graph.*, 28(3):30:1–30:10, July 2009. ISSN 0730-0301. doi: 10.1145/1531326.1531336. URL http://doi.acm.org/10.1145/1531326.1531336.

Oskar Elek, Tobias Ritschel, Carsten Dachsbacher, and Hans-Peter Seidel. Interactive light scattering with principal-ordinate propagation. In *Proceedings of Graphics Interface 2014*, GI '14, pages 87–94, Toronto, Ont., Canada, Canada, 2014. Canadian Information Processing Society. ISBN 978-1-4822-6003-8. URL http://dl.acm.org/citation.cfm?id=2619648.2619663.

Thomas J. Farrell, Michael S. Patterson, and Brian Wilson. A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties *in vivo*. *Medical Physics*, 19(4):879–888, July/August 1992.

Raanan Fattal. Participating media illumination using light propagation maps. *ACM Transactions on Graphics*, 28(1):7:1–7:11, January 2009.

Alexander L. Fetter, John Dirk Walecka, and Physics. *Theoretical Mechanics of Particles and Continua (Dover Books on Physics)*. Dover Publications, 2003. ISBN 978-0486432618.

Roald Frederickx and Philip Dutré. A forward scattering dipole model from a functional integral approximation. *ACM Trans. Graph.*, 36(4), July 2017. doi: 10.1145/3072959.3073681. URL http://dx.doi.org/10.1145/3072959.3073681.

Jeppe Revall Frisvad, Rasmus Revall Frisvad, Niels Jørgen Christensen, and Peter Falster. Scene independent real-time indirect illumination. In *Proceedings of Computer Graphics International (CGI) 2005*, pages 185–190+275. IEEE, June 2005.

Jeppe Revall Frisvad, Toshiya Hachisuka, and Thomas Kim Kjeldsen. Directional dipole model for subsurface scattering. *ACM Transactions on Graphics*, 34(1):5:1–5:12, November 2014.

Pascal Gautron. *Practical Global Illumination with Irradiance Caching*. Morgan and Claypool Publishers, 2009. ISBN 1598296442, 9781598296440.

A. S. Glassner. Tutorial: Computer graphics; image synthesis. chapter Space Subdivision for Fast Ray Tracing, pages 160–167. Computer Science Press, Inc., New York, NY, USA, 1988. ISBN 0-8186-8854-4. URL http://dl.acm.org/citation.cfm?id=95075.95113.

Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph.*, 18(3):213–222, January 1984. ISSN 0097-8930. doi: 10.1145/964965.808601. URL http://doi.acm.org/10.1145/964965.808601.

Paul Green, Jan Kautz, Wojciech Matusik, and Frédo Durand. View-dependent precomputed light transport using nonlinear Gaussian function approximations. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, I3D '06, pages 7–14, New York, NY, USA, 2006. ACM. ISBN 1-59593-295-X. doi: 10.1145/1111411.1111413. URL http://doi.acm.org/10.1145/1111411.1111413.

Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Comput. Graph. Appl.*, 18(2):32–43, March 1998. ISSN 0272-1716. doi: 10.1109/38.656788. URL http://dx.doi.org/10.1109/38.656788.

Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. *Computer Graphics (Proceedings of ACM SIGGRAPH 93)*, pages 165–174, August 1993.

Vlastimil Havran. *Heuristic Ray Shooting Algorithms*. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000. URL http://www.cgg.cvut.cz/~havran/phdthesis.html.

Miloš Hašan, Fabio Pellacini, and Kavita Bala. Direct-to-indirect transfer for cinematic relighting. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 1089–1097, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: 10.1145/1179352.1141998. URL http://doi.acm.org/10.1145/1179352.1141998.

Kyle Hegeman, Michael Ashikhmin, and Simon Premože. A lighting model for general participating media. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D '05, pages 117–124, New York, NY, USA, 2005. ACM. ISBN 1-59593-013-2. doi: 10.1145/1053427.1053446. URL http://doi.acm.org/10.1145/1053427.1053446.

Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. Real-time polygonal-light shading with linearly transformed cosines. *ACM Trans. Graph.*, 35(4):41:1–41:8, July 2016. ISSN 0730-0301. doi: 10.1145/2897824.2925895. URL http://doi.acm.org/10.1145/2897824.2925895.

J. Hendrich, D. Meister, and J. Bittner. Parallel BVH construction using progressive hierarchical refinement. *Comput. Graph. Forum*, 36(2):487–494, May 2017. ISSN 0167-7055. doi: 10.1111/cgf.13143. URL https://doi.org/10.1111/cgf.13143.

L. G. Henyey and J. L. Greenstein. Diffuse radiation in the galaxy. *Annales d'Astrophysique*, 3:117–137, 1940. Also in *The Astrophysical Journal 93*, 1941.

Rama Karl Hoetzlein. GVDB: Raytracing sparse voxel database structures on the GPU. In *Proceedings of High Performance Graphics*, HPG '16, pages 109–117, Aire-la-Ville, Switzerland, Switzerland, 2016. Eurographics Association. ISBN 978-3-03868-008-6. doi: 10.2312/hpg.20161197. URL http://dx.doi.org/10.2312/hpg.20161197.

J Hooker. Volumetric global illumination at Treyarch. In *Advances in Real-Time Rendering in Games*, ACM SIGGRAPH 2016 Courses. 2016. URL http://advances.realtimerendering.com/s2016/.

Wenzel Jakob and Steve Marschner. Manifold exploration: A Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph.*, 31(4):58:1–58:13, July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185554. URL http://doi.acm.org/10.1145/2185520.2185554.

Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques '96*, pages 21–30. Springer-Verlag, 1996.

Henrik Wann Jensen and Juan Buhler. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002)*, 21(3):576–581, July 2002.

Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001*, pages 511–518. ACM, August 2001.

J. Jimenez, K. Zsolnai, A. Jarabo, C. Freude, T. Auzinger, X.-C. Wu, J. von der Pahlen, M. Wimmer, and D. Gutierrez. Separable subsurface scattering. *Computer Graphics Forum*, 2015. To appear.

James T. Kajiya. The rendering equation. *Computer Graphics (Proceedings of SIGGRAPH 86)*, 20(4):143–150, August 1986.

Malvin H. Kalos and Paula A. Whitlock. *Monte Carlo Methods*. Wiley-VCH, 2008. ISBN 978-3527407606.

Anton Kaplanyan. Light propagation volumes in CryEngine 3. In *ACM SIGGRAPH 2009 Talks, Advanced Real-time Rendering Course.*, 2009.

Brian Karis. High-quality temporal supersampling. In *Advances in Real-Time Rendering in Games, Part I*, number 10 in ACM SIGGRAPH 2014 Courses. 2014. URL http://advances.realtimerendering.com/s2014/.

Tero Karras and Timo Aila. Fast parallel construction of high-quality bounding volume hierarchies. In Kayvon Fatahalian and Christian Theobalt, editors, *Proceedings of High Performance Graphics (HPG '13)*, pages 89–99. ACM, 2013. ISBN 978-1-4503-2135-8. doi: 10.1145/2492045.2492055.

Alexander Keller. Instant radiosity. In *Proceedings of ACM SIGGRAPH 97*, pages 49–56, 1997.

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. Gradient-domain path tracing. *ACM Trans. Graph.*, 34(4):123:1–123:13, July 2015. ISSN 0730-0301. doi: 10.1145/2766997.

David Kirk and James Arvo. Unbiased sampling techniques for image synthesis. *SIGGRAPH Comput. Graph.*, 25(4):153–156, July 1991. ISSN 0097-8930. doi: 10.1145/127719.122735. URL http://doi.acm.org/10.1145/127719.122735.

Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. In *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, pages 75:1–75:19, New York, NY, USA, 2008. ACM. doi: 10.1145/1401132.1401228. URL http://doi.acm.org/10.1145/1401132.1401228.

Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 31–42, New York, NY, USA, 1996. ACM. ISBN 0-89791-746-4. doi: 10.1145/237170.237199. URL http://doi.acm.org/10.1145/237170.237199.

Dongping Li, Xin Sun, Zhong Ren, Stephen Lin, Yiying Tong, Baining Guo, and Kun Zhou. TransCut: Interactive rendering of translucent cutouts. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):484–494, March 2013.

M. Mara, M. McGuire, D. Nowrouzezahrai, and D. Luebke. Deep G-buffers for stable global illumination approximation. In *Proceedings of High Performance Graphics*, HPG '16, pages 87–98, Aire-la-Ville, Switzerland, Switzerland, 2016. Eurographics Association. ISBN 978-3-03868-008-6. doi: 10.2312/hpg.20161195. URL http://dx.doi.org/10.2312/hpg.20161195.

Michael Mara, David Luebke, and Morgan McGuire. Toward practical real-time photon mapping: Efficient GPU density estimation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*,

I3D '13, pages 71–78, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1956-0. doi: 10.1145/2448196.2448207. URL http://doi.acm.org/10.1145/2448196.2448207.

Michael Mara, Morgan McGuire, Benedikt Bitterli, and Wojciech Jarosz. An efficient denoising algorithm for global illumination. In *Proceedings of High Performance Graphics*, HPG '17, pages 3:1–3:7, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5101-0. doi: 10.1145/3105762.3105774. URL http://doi.acm.org/10.1145/3105762.3105774.

Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, 22(3):759–769, 2003.

Morgan McGuire and Michael Mara. Efficient GPU screen-space ray tracing. *Journal of Computer Graphics Techniques (JCGT)*, 3(4):73–85, December 2014. ISSN 2331-7418. URL http://jcgt.org/published/0003/04/04/.

Morgan McGuire, Mike Mara, Derek Nowrouzezahrai, and David Luebke. Real-time global illumination using precomputed light field probes. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '17, pages 2:1–2:11, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4886-7. doi: 10.1145/3023368.3023378.

Soham Uday Mehta, Brandon Wang, Ravi Ramamoorthi, and Fredo Durand. Axis-aligned filtering for interactive physically-based diffuse indirect lighting. *ACM Trans. Graph.*, 32(4):96:1–96:12, July 2013. ISSN 0730-0301. doi: 10.1145/2461912.2461947. URL http://doi.acm.org/10.1145/2461912.2461947.

D. Meister and J. Bittner. Parallel locally-ordered clustering for bounding volume hierarchy construction. *IEEE Transactions on Visualization and Computer Graphics*, 24(3):1345–1353, March 2018. ISSN 1077-2626. doi: 10.1109/TVCG.2017.2669983.

Tom Mertens, Jan Kautz, Philippe Bekaert, Hans-Peter Seidel, and Frank Van Reeth. Interactive rendering of translucent deformable objects. In *Proceedings of Eurographics Symposium on Rendering (EGSR 2003)*, pages 130–140, June 2003.

Oliver Nalbach, Tobias Ritschel, and Hans-Peter Seidel. Deep screen space. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '14, pages 79–86, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2717-6. doi: 10.1145/2556700.2556708. URL http://doi.acm.org/10.1145/2556700.2556708.

F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. Technical report, National Bureau of Standards (US), October 1977.

Mangesh Nijasure, Sumanta Pattanaik, and Vineet Goel. Real-time global illumination on GPUs. *Journal of Graphics Tools*, 10(2):55–71, 2005. doi: 10.1080/2151237X.2005.10129194. URL https://doi.org/10.1080/2151237X.2005.10129194.

Jeffry S. Nimeroff, Eero Simoncelli, and Julie Dorsey. Efficient re-rendering of naturally illuminated environments. In *Photorealistic Rendering Techniques (Proceedings of EGWR 1994)*, page 359–373, 06/1994 1994.

Georgios Papaioannou. Real-time diffuse global illumination using radiance hints. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, HPG '11, pages 15–24, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0896-0. doi: 10.1145/2018323.2018326. URL http://doi.acm.org/10.1145/2018323.2018326.

S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M Stich. OptiX: A general purpose ray tracing engine. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)*, 29(4):66:1–66:13, 2010.

Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)*, 35(6):179:1–179:12, November 2016.

Fabio Pellacini, Kiril Vidimče, Aaron Lefohn, Alex Mohr, Mark Leone, and John Warren. Lpics: A hybrid hardware-accelerated relighting engine for computer cinematography. *ACM Trans. Graph.*, 24(3):464–470, July 2005. ISSN 0730-0301. doi: 10.1145/1073204.1073214. URL http://doi.acm.org/10.1145/1073204.1073214.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann/Elsevier, third edition, 2017.

R W. Preisendorfer. *Radiative Transfer on Discrete Spaces*. Pergamon Press, 1965. ISBN 978-0-08-010592-5.

R.W. Preisendorfer. *Hydrologic optics*. U.S. Dept. of Commerce, National Oceanic and Atmospheric Administration, Environmental Research Laboratories, Pacific Marine Environmental Laboratory, 1976.

Simon Premože, Michael Ashikhmin, and Peter Shirley. Path integration for light transport in volumes. In *Proceedings of the 14th Eurographics Workshop on Rendering*, EGRW '03, pages 52–63, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. ISBN 3-905673-03-7. URL http://dl.acm.org/citation.cfm?id=882404.882413.

T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph.*, 27(5):129:1–129:8, December 2008. ISSN 0730-0301. doi: 10.1145/1409060.1409082. URL http://doi.acm.org/10.1145/1409060.1409082.

Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. The state of the art in interactive global illumination. *Comput. Graph. Forum*, 31 (1):160–188, February 2012. ISSN 0167-7055. doi: 10.1111/j.1467-8659.2012.02093.x. URL http://dx.doi.org/10.1111/j.1467-8659.2012.02093.x.

Holly Edith Rushmeier. *Realistic image synthesis for scenes with radiatively participating media*. PhD thesis, Cornell University, Ithaca, NY, USA, 1988.

Szymon Rusinkiewicz. A new change of variables for efficient BRDF representation. In *Eurographics Rendering Workshop*, pages 11–22, 1998.

Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. *SIGGRAPH Comput. Graph.*, 24(4):197–206, September 1990. ISSN 0097-8930. doi: 10.1145/97880.97901. URL http://doi.acm.org/10.1145/97880.97901.

Daniel Scherzer, Chuong H. Nguyen, Tobias Ritschel, and Hans-Peter Seidel. Pre-convolved radiance caching. *Comput. Graph. Forum*, 31(4):1391–1397, June 2012. ISSN 0167-7055. doi: 10.1111/j.1467-8659.2012.03134.x. URL http://dx.doi.org/10.1111/j.1467-8659.2012.03134.x.

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. Spatiotemporal variance-guided filtering: Real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*, HPG '17, pages 2:1–2:12, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5101-0. doi: 10.1145/3105762.3105770. URL http://doi.acm.org/10.1145/3105762.3105770.

Mikio Shinya, Yoshinori Dobashi, Michio Shiraishi, Motonobu Kawashima, and Tomoyuki Nishita. Multiple scattering approximation in heterogeneous media by narrow beam distributions. *Comput. Graph. Forum*, 35(7): 373–382, October 2016. ISSN 0167-7055. doi: 10.1111/cgf.13034. URL https://doi.org/10.1111/cgf.13034.

Peter Shirley, Changyaw Wang, and Kurt Zimmerman. Monte Carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, January 1996.

Ari Silvennoinen and Jaakko Lehtinen. Real-time global illumination by precomputed local reconstruction from sparse radiance probes. *ACM Trans. Graph.*, 36(6):230:1–230:13, November 2017. ISSN 0730-0301. doi: 10.1145/3130800.3130852. URL http://doi.acm.org/10.1145/3130800.3130852.

Florian Simon, Johannes Hanika, and Carsten Dachsbacher. Rich-VPLs for improving the versatility of many-light methods. *Comput. Graph. Forum*, 34(2):575–584, May 2015. ISSN 0167-7055. doi: 10.1111/cgf.12585. URL http://dx.doi.org/10.1111/cgf.12585.

Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, 21(3):527–536, July 2002. ISSN 0730-0301. doi: 10.1145/566654.566612. URL http://doi.acm.org/10.1145/566654.566612.

Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, 22(3):382–391, July 2003.

Jos Stam. Multiple scattering as a diffusion process. In P. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*, pages 41–50. Springer, June 1995. Proceedings of the Sixth Eurographics Workshop on Rendering.

Jonathan Dyssel Stets, Alessandro Dal Corso, Jannik Boll Nielsen, Rasmus Ahrenkiel Lyngby, Sebastian Hoppe Nesgaard Jensen, Jakob Wilm, Mads Brix Doest, Carsten Gundlach, Eythor Runar Eiriksson, Knut Conradsen, Anders Bjorholm Dahl, Jakob Andreas Bærentzen, Jeppe Revall Frisvad, and Henrik Aanæs. Scene reassembly after multimodal digitization and pipeline evaluation using photorealistic rendering. *Appl. Opt.*, 56(27):7679–7690, Sep 2017. doi: 10.1364/AO.56.007679. URL http://ao.osa.org/abstract.cfm?URI=ao-56-27-7679.

George Gabriel Stokes. *On the perfect Blackness of the Central Spot in Newton's Rings, and on the Verification of Fresnel's Formula for the intensities of Reflected and Reflacted Rays*, volume 2 of *Cambridge Library Collection - Mathematics*, page 89–103. Cambridge University Press, 2009. doi: 10.1017/CBO9780511702259.008.

T. Tanaka, A. Fujimoto, and K. Iwata. ARTS: accelerated ray-tracing system. *IEEE Computer Graphics and Applications*, 6:16–26, 04 1986. ISSN 0272-1716. doi: 10.1109/MCG.1986.276715. URL doi.ieeecomputersociety.org/10.1109/MCG.1986.276715.

Parag Tole, Fabio Pellacini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. *ACM Trans. Graph.*, 21(3):537–546, July 2002. ISSN 0730-0301. doi: 10.1145/566654.566613. URL http://doi.acm.org/10.1145/566654.566613.

Kostas Vardis, Georgios Papaioannou, and Anastasios Gkaravelis. Real-time radiance caching using chrominance compression. *Journal of Computer Graphics Techniques (JCGT)*, 3(4):111–131, December 2014. ISSN 2331-7418. URL http://jcgt.org/published/0003/04/06/.

Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of ACM SIGGRAPH 1995*, pages 419–428, 1995.

Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of ACM SIGGRAPH 1997*, pages 65–76, 1997.

Ingo Wald, Sven Woop, Carsten Benthin, Gregory S. Johnson, and Manfred Ernst. Embree: A kernel framework for efficient CPU ray tracing. *ACM Trans. Graph.*, 33(4):143:1–143:8, July 2014. ISSN 0730-0301. doi: 10.1145/2601097.2601199. URL http://doi.acm.org/10.1145/2601097.2601199.

Bruce Walter, George Drettakis, and Donald P. Greenberg. Enhancing and optimizing the render cache. In *Proceedings of the Eurographics Workshop on Rendering (EGWR 2002)*, pages 37–42. ACM Press, 2002.

Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: A scalable approach to illumination. *ACM Trans. Graph.*, 24(3):1098–1107, July 2005. ISSN 0730-0301. doi: 10.1145/1073204.1073318. URL http://doi.acm.org/10.1145/1073204.1073318.

Rui Wang, Ewen Cheslack-Postava, Rui Wang, David Luebke, Qianyong Chen, Wei Hua, Qunsheng Peng, and Hujun Bao. Real-time editing and relighting of homogeneous translucent materials. *The Visual Computer*, 24(7):565–575, July 2008.

Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph.*, 22(4):85–92, June 1988. ISSN 0097-8930. doi: 10.1145/378456.378490. URL http://doi.acm.org/10.1145/378456.378490.

S. Widmer, D. Pająk, A. Schulz, K. Pulli, J. Kautz, M. Goesele, and D. Luebke. An adaptive acceleration structure for screen-space ray tracing. In *Proceedings of the 7th Conference on High-Performance Graphics*, HPG '15, pages 67–76, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3707-6. doi: 10.1145/2790060.2790069. URL http://doi.acm.org/10.1145/2790060.2790069.

Chris Wylie, Gordon Romney, David Evans, and Alan Erdahl. Half-tone perspective drawings by computer. In *Proceedings of the November 14-16, 1967, Fall Joint Computer Conference*, AFIPS '67 (Fall), pages 49–58, New York, NY, USA, 1967. ACM. doi: 10.1145/1465611.1465619. URL http://doi.acm.org/10.1145/1465611.1465619.